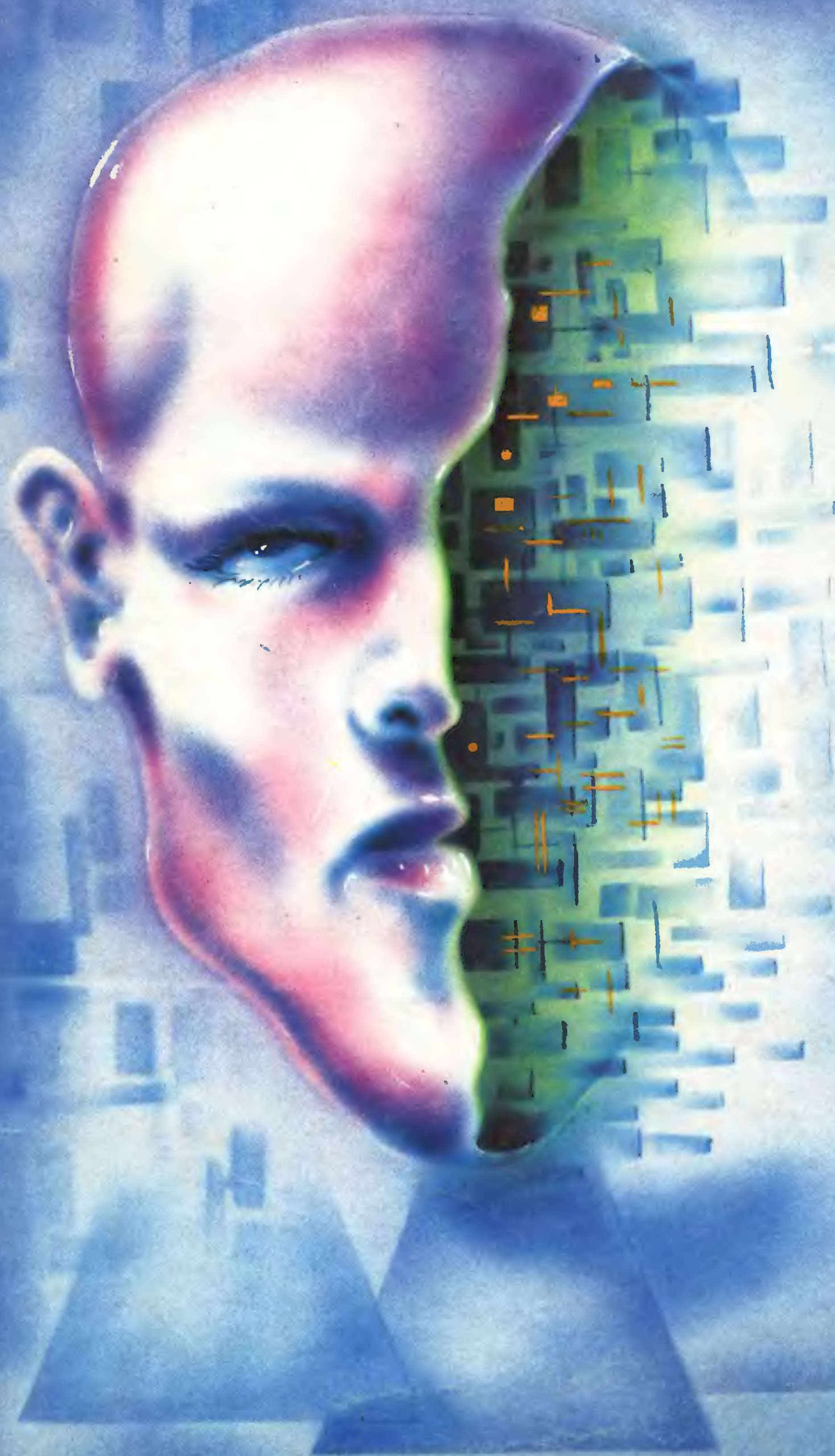
Commodore

Cena 10 tys. zł
nr indeksu 355275

6/92

KEMAN

Miesięcznik Użytkowników Komputerów C-64 i Amiga



Commodore



nr indeksu: 355275

Wydawca:
KEBAB - sp. z o.o.
ul. Wojciechowskiego 28
PL - 71 476 Szczecin
telefon: (091)77674
telefax: (091)45402

Redaguje kolegium w składzie:
Krzysztof Kobus, Patryk Łogiewa, Grzegorz Mikuła, Krzysztof Moroń, Paweł Sołtysiński

Prezes zarządu Spółki:
Piotr Sołtysiński

Redaktor naczelny:
Patryk Łogiewa

Szef działu AMIGA:
Krzysztof Kobus
tel.: (091)525336

Szef działu C-64:
Paweł Sołtysiński
tel.: (091)77674

Stale obecni na łamach:
Robert Turliński
Miłosław Smyk
Arkadiusz Zych

Redakcja nie zwraca nie zamówionych materiałów, oraz zastrzega sobie prawo wprowadzenia zmian w otrzymanych rękopisach

Projekt okładki:
Dariusz Zawadzki

Zdjęcia:
Sławomir Borek/„Panorama"

W sprawie kolportażu...

I TY MOŻESZ BYĆ KOLPORTEREM "KEBABA"!

Chcielibyśmy zainteresować naszą propozycją te osoby fizyczne i prawne, które chciałyby się podjąć kolportażu we własnym zakresie. Zapewnimy jednocześnie co najmniej tygodniowe wyprzedzenie przed przekazaniem nakładu do dystrybucji przez krajowego kolportera. Zapraszamy do współpracy studia komputerowe, księgarnie i obrotne osoby indywidualne. Szczegółowych informacji idziela się telefonicznie i w siedzibie redakcji.

... reklamy ...

Przedsiębiorstwo KEBAB spółka z o.o. oferuje Państwu szybką i tanią obsługę reklamową. Ogłoszenia drobne od osób indywidualnych (do 10 słów na wyciętym z numeru kuponie) przyjmujemy bezpłatnie. Większe - 1000 zł za słowo. Reklamy ramkowe (minimalny format - 20 cm kwadratowych): 1 cm² - 4500 zł; cała strona - 2,5 miliona zł; dodatkowy kolor - odpowiednio 50% drożej. „Ogłoszenia na IV stronie okładki: cała - 4 miliony zł; 1/2 - 2,5 miliona zł". Treść ogłoszeń przyjmujemy za pośrednictwem poczty (adres - patrz stopka redakcyjna) lub „Media Banco Brocker" Sp. z o.o.; 02-555 Warszawa, Al. Niepodległości 177; tel. 253 684; fax. 257 899. Ogłoszenia wraz z określeniem formatu reklamy prosimy nadsyłać listem poleconym. Dołączenie odcinka wpłaty znacznie przyspieszy zamieszczenie ogłoszenia.

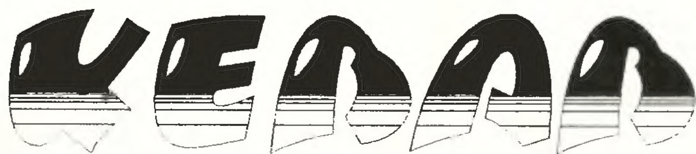
... i prenumeraty.

Aby uporać się z problemem ciągłego wzrostu cen usług poligraficznych, papieru itp. i uniknąć dokonywania przez Czytelników Kłopotliwych dopłat, postanowiliśmy wprowadzić tzw. małą prenumeratę w okresach 3-miesięcznych. Rozwiązanie to gwarantuje Czytelnikom niezmienność ceny w okresie, który obejmuje zamówienie. Cenę egzemplarza wraz z kosztem usługi pocztowej skalkulowaliśmy na 9500 zł. Daje to następujące możliwości:

numery 7, 8, 9 - 28 500,-

KEBAB sp. z o.o.
Pomorski Bank Kredytowy II Oddział w Szczecinie
konto nr: 368113-25771-136

Należy również podać DOKŁADNY ADRES, IMIĘ I NAZWISKO zamawiającego!



Nr 6 Czerwiec 1992

Digi Tiger II
to najszybszy vdeo digitizer dla Amigi spośród nie pracujących w czasie rzeczywistym !

Baza danych 64
to prosty (choć wcale nie prymitywny) program przeznaczony dla użytkowników C-64

PC-Task V1.04
to nowy, programowy emulator IBM PC
opisu szukaj na stronie 9

Żarłoczny robaczek !
to gra, w BASIC'u, którą warto "wklepać"

Grafika wektorowa
cóż więcej dodać? Szukaj na str. 19

Lubisz pograć ?
W coś dobrego ? Dzisiaj masz sporo lektury!
Zacznij jak zwykle od strony 24

Spis treści:

















Nowości z kraju i ze świata...

Separacja barw to proces wykorzystywany w poligrafii podczas przygotowywania kolorowych materiałów do druku. W przypadku komputerów oznacza ona konwersję grafiki i tekstu z formatu RGB (lub innego używanego przy wyświetlaniu obrazu na monitorze) na CMYK celem właściwego odwzorowania barw na papierze. I tu ciekawostka. Otóż z notatki „Ventura dla Windows" zamieszczonej w majowym numerze „Bajtka" dowiadujemy się, iż: *„Dotychczas (tzn. do momentu pojawienia się pakietu Ventura 4.0 for Windows - przyp. KEBAB'a) tylko programy pracujące na komputerach Macintosh potrafiły dokonać tzw. separacji kolorów (...)"*. Bardzo nam miło, że teraz także na pecetach będzie można robić DTP z prawdziwego zdarzenia, niemniej uważamy za celowe napomknąć, iż dla Amigi już dłuższy czas istnieją programy umożliwiające przeprowadzanie wspomnianej separacji na poziomie profesjonalnym - ot, choćby Art Department Professional, PageStream, Saxon Publisher czy Professional Page, i że w swej fascynacji „prawdziwymi" (w przeciwieństwie do „zabawkowej" Amigi) komputerami nie należy o tym zapominać, a co za tym idzie - dezinformować Czytelników!

Po ponad rocznej ciszy w temacie „tracker'y" pojawił się wreszcie w kwietniu tego roku produkt Petera Hanninga z grupy Noxious - ProTracker 2.1a. Program bazuje na kodzie źródłowym wersji 2.0 autorstwa grupy Amiga Freelancers, i jak nietrudno się domyślić jest jej rozwinięciem. Większość zmian ma jednak charakter wyłącznie kosmetyczny, ze znaczniejszych zaś warto wymienić współpracę z biblioteką powerpacker.library pozwalającą na operowanie danymi (modułami i samplami) w skompresowanej formie, opcję odczytu i zapisu pojedynczych ścieżek i pattern'ów oraz możliwość podania nazwy modułu do załadowania po uruchomieniu programu jako argumentu w CLI lub Workbench'u. Usprawniono również komunikację z użytkownikiem i usunięto wiele drobnych błędów, czyniąc niestety kilka nowych. Najbardziej rzucają się w oczy kłopoty z zarządzaniem pamięcią uniemożliwiające poprawną pracę programu na komputerach pozbawionych SLOW lub FAST RAM-u. Dyskusyjny jest również bardzo pamięciożerny sposób ładowania skompresowanych danych - odbywający się za pośrednictwem ramdysku i wymagający co najmniej dwukrotnie większej ilości wolnej pamięci niż zajmuje aktualnie ładowany modul/sampling. Ponadto, jak sam przyznaje twórca programu, nie był on zbyt wnikliwie testowany pod kontrolą systemu KickStart 2.0, nie należy się zatem dziwić ewentualnym „kwiatkom" mogącym ujawnić się przy pracy z nowym ROMem. Z lektury dokumentacji dołączonej do ProTrackera 2.1a można wnieść iż autor pełen jest zapału do pracy, co pozwala spodziewać się kolejnych, „odpluskwionych" wersji programu w niedalekiej przyszłości. A ma się w nich pojawić między innymi pełna obsługa MIDI, syntetyczne dźwięki, współpraca z Intuition (a więc pełny multitasking) itd. Pozostaje tylko czekać...

Z kolei firma Electronic Arts wypuściła niedawno upgrade do programu Deluxe Paint IV. Najnowsza wersja tego niezwykle popularnego pakietu graficznego nosząca numer 4.1 jest odpowiedzią firmy na uwagi i sugestie zgłaszane przez użytkowników i beta-testerów (osoby zajmujące się testowaniem oprogramowania jeszcze przed jego wypuszczeniem na rynek, ale już po alfa-testingu dokonywanym w pracowniach autorów), jest zatem jeszcze wygodniejsza w obsłudze, nieco szybsza (w trybie HAM) i bardziej dostosowana do Kickstart'u 2.0. Szczegółów szukajcie w następnym Kebabie.

Wydaje się, że w dziedzinie produkcji sprzętowych rozszerzeń przeznaczonych do Amigi będziemy już wkrótce mieli światowego monopolistę. Wcale jednak nie będzie nim firma Commodore. GVP - tak prawdopodobnie brzmi hasło przyszłości Amigi. Firma ta, po niemal całkowitym zdominowaniu amerykańskiego rynku "Amigowców", stawia coraz śmielsze kroki w Europie. Nie ma zresztą czemu się dziwić. Najwyższej klasy technologia w połączeniu z doskonałymi konstrukcjami oraz w niczym im nie ustępującym oprogramowaniem, powoduje, że niemal każdy produkt określany przymiotnikami "najlepszy", "najszybszy" itp. posiada plakietkę Great Valley Products. G-Force 030 "combo", G-Force 040, Impact Vision 24 czy Digital Sound Studio to tylko najbardziej przebojowe produkty spod znaku "Wielkiej Doliny". Firma nie ogranicza się jednak do samodzielnego konstruowania. Porozumienie z niemiecką firmą Vortex Computersysteme GmbH doprowadziło do powstania "GVP PC286-16", czyli inaczej mówiąc ATonce plus w wersji dostosowanej do współpracy z innymi urządzeniami GVP. Przykładowo ATonce w wersji GVP instaluje się nie jak w przypadku oryginalnego Vortex'a w gniazdo procesora, lecz umieszcza się go w obudowie twardego dysku dołączanego do A500 z boku. Rozwiązanie takie gwarantuje kompatybilność z kontrolerem naszego "twardziela" oraz pozostawia zdecydowanie więcej luzu wewnątrz i tak dosyć ciasnej obudowy standardowej Amigi 500.

Znana ze swoich produktów serii „Pro" firma Gold Disk wypuściła niedawno oprócz kolejnej wersji programu DTP - Professional Page V3.0 także pakiet kalkulacyjny o nazwie Professional Calc. Niestety specjaliści od marketingu firmy Gold Disk zdecydowali się na skuteczne ograniczenie sprzedaży tego skądinąd bardzo dobrego programu ustalając cenę na bardzo wysokim poziomie. Przykładowo około 600,- DM w Niemczech.

Firma Commodore wprowadziła na rynek kolejną kartę z serii Bridgeboard. Karta nazywa się A2386SX i jest emulatorem komputera serii AT386SX z zegarem 20 MHz.

Również znana z emulatorów ATonce firma Vortex produkuje już kartę emulacyjną AT386SX dla Amigi.

Miłośników komputerowych efektów video ucieszy z pewnością fakt pojawienia się na rynku programu do cyfrowej obróbki obrazu o nazwie Image Master. Pakiet ten działający na podobnych zasadach co znany już od dłuższego czasu „Art Department" posiada jednak sporo nowych możliwości. Jako jedną z najbardziej fascynujących należy wspomnieć funkcję „morphing". Przykłady działania takiej funkcji mieliśmy możliwość oglądać w legendarnym już Terminatorze 2. Co prawda nie oznacze to, że przy użyciu Amigi uda nam się stworzyć od razu Terminatora 3, ale jakość efektów uzyskiwanych przy użyciu tej funkcji Image Master'a jest według czasopism zachodnich niewiarygodnie wysoka. Program nie jest konkurencją dla ADPro, lecz jego doskonałym uzupełnieniem.

W trzecim kwartale bieżącego roku firma Fujitsu ma rozpocząć seryjną produkcję przedstawionych na targach CEBIT 92 napędów magneto-optycznych dysków trzy i pół calowej średnicy. Dyski takie przypominające z wyglądu normalną dyskietkę 3.5 potrafią aktualnie pomieścić 128MB danych. Dzięki zastosowaniu standardowego łącza SCSI-2 nie powinno być problemów z podłączeniem takiego napędu do Amigowego kontrolera.

Kolejne karty grafiki 24-o bitowej dla Amigi zostały wprowadzone na rynek. „Rembrandt" oraz „Rainbow" to karty, które potrafią wyświetlić na ekranie obraz w 1310720 kolorach jednocześnie ! Ktoś mógłby powiedziec: No i co z tego? Przecież karty 24-o bitowe potrafią pokazać 16777216 kolorów jednocześnie... I tu tkwi poważny błąd logiczny, na który dali się złapać swego czasu również nasi poważni dealerzy od Macintosh'ów. Otóż jeżeli nawet każdy punkt na ekranie będzie miał inną barwę od wszystkich pozostałych, to i tak przy rozdzielczości 640x512 zobaczymy (jeżeli nam się uda) „zaledwie" 327680 kolorów. Przy rozdzielczości 1280x1024 oferowanej przez nowe karty, liczba ta wzrasta do wspomnianego już miliona trzystu dziesięciu tysięcy...

Znana z różnych ciekawostek firma Rossmoeller ma zamiar wprowadzić na rynek prawdziwą niespodziankę. Ma to być turbo karta, czyli inaczej mówiąc „dopalacz" lub „coś z kopem", przeznaczona dla... Commodore 64. Przypomnijmy, że firma ta produkowała już swego czasu podobne urządzenie, jednakże wysoka cena oraz spora niekompatybilność spowodowała, że „Turbo Process" nie zyskał zbyt wielkiej popularności. Tym razem ma być inaczej. Karta ma obsługiwać do 4MB (!) RAM, a zainstalowany na niej procesor 65816 ma być taktowany z częstotliwością 8MHz (!) Powinno to dać efekty w postaci sześć do osiem razy większej szybkości obliczeń. Firma zapewnia przy tym, że tym razem zostanie zachowana wysoka kompatybilność z oryginalnym systemem operacyjnym C-64. Aby przy tych wszystkich superlatywach nie trzeba było „wiekami" czekać na operacje dyskowe, zostanie również zainstalowany superszybki system operacji dyskowych na wzór Dolphin- czy Prologic-Dos.

Inną niespodzianką będzie wypuszczenie na rynek emulatora C-64 przeznaczonego dla komputera... Acorn Archimedes. Związane jest to prawdopodobnie z polityką rynkową firmy Acorn, która po niepowodzeniu swojego produktu na rynku profesjonalnym stara się aktualnie wprowadzić Archimedesa jako komputer dla byłych posiadaczy C-64. Dla Czytelników, którzy nie słyszeli nigdy nic o tym komputerze kilka wyjaśnień. Acorn Archimedes to szybki komputer zbudowany w oparciu o procesor tzw. typu RISC. Spore możliwości graficzne do 640x512 i 1056x256 punktów w 256 kolorach jednocześnie, paleta 4096 kolorów oraz system operacyjny umożliwiający multitasking, to cechy, które zbliżają go do Amigi. Natomiast szybkość obliczeń jest w przypadku 32-bitowych procesorów RISC znacznie większa. Przykładowy test napisany (w BASIC'u) potrzebował do wykonania się na C-64 ponad cztery godziny, na Amidze (Amiga BASIC) już "tylko" 55 minut. Natomiast BASIC znajdujący się w ROM'ie Archimedesa A3000 zajmował się tym samym testem przez 3 minuty i 17 sekund (dane liczbowe za M&T 64'er). Dźwięk jest przekazywany poprzez osiem niezależnych kanałów stereo

SD!

Quo Vadis ?

komputerze...

Komputery. Zastępują człowieka tam gdzie jego szybkość myślenia i zdolność do zapamiętywania okazują się niewystarczające wobec wymogów współczesnej cywilizacji. To znaczy prawie wszędzie. Ilość dziedzin życia, których jeszcze nie opanowały maleje z dnia na dzień w postępie geometrycznym. Spotykamy je na każdym kroku - ciche, usłużne, szybkie, na ogół nie dające powodów do narzekań. Częstokroć niezauważalne. Gdy jednak z ich pomocą człowiek wykona kolejny krok w stronę realizacji swoich marzeń, znajdują się na ustach wszystkich. Takim krokiem w dziedzinie techniki filmowej był „Terminator 2"...

Komputery zaczęto wykorzystywać w kinematografii kiedy tylko okazało się, że rozdzielczość ich grafiki oraz paleta barw nadają się do tego celu. Poczynając od sławetnego TRON'a, poprzez takie filmy jak The Last Starfigher, gdzie wszystkie pojazdy i sekwencje ich walki w kosmosie zrealizowano wyłącznie przy użyciu komputerów, czy Young Sherlock Holmes (w polskich kinach pod tytułem Piramida Strachu) z robiącym niesamowite wrażenie, wyłaniającym się z witraża rycerzem, twórcy efektów specjalnych wspinali się na kolejne stopnie trudnej sztuki omamienia widza. Jedna szczególnie pracownia - Industrial Light & Magic - osiągnęła na tym polu spore sukcesy. W ILM wykreowano, jeszcze technikami konwecjonalnymi, tak znane postaci jak na przykład Darth Vader czy E.T. Teraz, korzystając z pomocy 340 terminali Silicon Graphics VGX firma szokuje nas komputerowym ogniem (Backdraft) i wodą (The Abbys), których realistyczna animacja do niedawna zakrawała na niemożliwość. Również za stworzeniem T1000, mogącego przybrać dowolny kształt robota z filmu „Terminator 2" stoją ludzie z ILM, a konkretnie zespół poprowadzony przez Steve'a Williamsa. On sam zresztą twierdzi: „Efekty filmowe były takie same przez sto lat i zmieniają się tego roku". Brak skromności? Nie wydaje się, szczególnie po obejrzeniu linoleum zmieniającego się w człowieka.

W produkcję T2 zainwestowano ogromną sumę stu milionów dolarów, zaś największą część tej kwoty, oprócz honorarium Arnolda Schwarzenegger'a wynoszącego 12 mln., pochłonęły właśnie efekty specjalne. Pierwszym etapem stworzenia T1000 było dokładne przeniesienie wymiarów oraz sposobu poruszania się Roberta Patricka - aktora, który dał cyborgowi jego ludzką postać - do pamięci komputera. Czekało to również wszystkie obiekty docelowe, tak więc między innymi pilot śmigłowca, opiekunka Johna i strażnik szpitalny zostali przy użyciu techniki skanningu laserowego przetworzeni na reprezentację wire-form (na ekranie wygląda ona jak druciana siatka - stąd nazwa), skądinąd znaną użytkownikom programów do ray-tracing'u Do obliczenia faz pośrednich użyto algorytmu zwanego True Infinite Morphing, który nawiasem mówiąc staje się coraz bardziej popularny (umieszczono go np. w programie Image Master dla Amigi). Potem należało już „tylko" połączyć wygenerowane komputerowo postaci ze scenografią (w istocie był to najtrudniejszy fragment operacji - Czytelnikom bardziej obeznanym z programowaniem i technikami ray-tracing'u polecamy rozważenie algorytmu, który stworzył odbicie otoczenia na metalicznym korpusie T1000) i przeliczyć, korzystając z matematycznie opisanych praw fizyki.

Do obliczeń użyto trzydziestu sprzęgniętych w sieć minikomputerów, dzięki czemu dało się zredu-

GYROS!



kować czas potrzebny na ich przeprowadzenie do niezbędnego minimum. Koszty - jedna ramka około $100, minuta filmu - $150.000, jakość - to po prostu trzeba zobaczyć.
Obecnie, po olbrzymim sukcesie Terminatora 2, ILM pracuje nad efektami do filmu Stevena Spielberga Jurassic Park, w którym pojawiają się dinozaury w interakcji z postaciami ludzkimi. Jako, że Spielberg ma bardzo wysokie wymagania co do realizmu obrazu, w użyciu będą komputery w połączeniu z metodami tradycyjnymi. Jeden z animatorów ILM miał rzekomo powiedzieć, iż będzie to projekt „dziesięciokrotnie trudniejszy niż stworzenie robota T1000", co jednak nie przeszkodziło firmie podjąć się owego zadania.
Wobec niezwykle dynamicznego rozwoju tej nowej dziedziny sztuki (przypomnijmy, iż metamorfoza (ang. morphing) po raz pierwszy pojawiła się w filmie Willow z 1988 roku) rodzi się pytanie jak będą wyglądały efekty specjalne za pięć, dziesięć lat? Williams mówi: „Dawno zmarli prezydenci pojawią się, komputerowo wygenerowani w telewizji, dając przemówienia. Aktorzy, którzy umarli 50 lat temu będą występować obok aktorów współczesnych. Moglibyśmy nawet stworzyć takich, którzy nigdy się nie narodzili (...)". Możliwości są jak więc widać ogromne, niewątpliwie jednak aż taki rozwój technik animacji komputerowej pociągnie za sobą szereg dyskusji natury etyczno-moralnej, wynikających z możliwości manipulowania opinią publiczną. To jest jednak temat na odrębny artykuł, ten zaś zakończmy słowami Williamsa: „W końcu jest to jeszcze jeden rodzaj ołówka. Jeśli jest w rękach kogoś, kto nie potrafi rysować, to sam z siebie też nic nie stworzy."

Miłosław „Thorgal" Smyk

W artykule wykorzystano materiały z „Time'a".

64

Baza Danych

dla Commodore 64

Dziś w KEBABIE coś dla miłośników programów w j. BASIC - prosta BAZA DANYCH. No cóż, działać to to działa ale ze względu na ograniczone zasoby pamięci Commodore 64, będzie to najwyżej spis telefonów lub książek w biblioteczce. Od czegoś jednak trzeba zaczynać. Pewnie nie wszyscy wiedzą, co to jest baza danych. Tym terminem zwykło się określac pewien zbiór zapisów o określonej strukturze, który umożliwia łatwy dostęp do danych. Mogą one być wyszukiwane na podstawie zadanego klucza (np. proszę wyświetlić mi na ekranie wszystkie tytuły książek, które pożyczyłem mojemu koledze, itp.). Aby w ten sposób zapisywać i operować danymi, stworzono wiele programów przeznaczonych do tych celów i oto jeden z nich czeka na chwilę cierpliwości przy jego wpisywaniu.
Może opiszmy obsługę tego programu na przykładzie naszego księgozbioru. Ile danych potrzebujemy, żeby opisać książkę? Dwóch, trzech czy pięciu? Może przyjmijmy taki układ:

1.Tytuł
2.Autor
3.Wydawnictwo
4.Komu pożyczono

Myślę, że powinno wystarczyć. Po uruchomieniu programu wybieramy opcję pierwszą (zakładanie bazy danych). Komputer pyta nas o nazwę dla naszych informacji więc podajemy mu np. „biblioteka". Następnie podajemy mu ilość kluczy opisujących każdy zapis (każdą książkę) - w naszym przypadku 4, a następnie wprowadzamy ich nazwy (tytuł, autor, itd.). Po powrocie do głównego menu i wybraniu opcji 4 (operacje na zapisach) komputer prosi nas o podanie pierwszego zapisu. Dlaczego? Dlatego, iż warunkiem operowania na jakichś danych jest posiadanie choćby jednego zapisu - jeżeli go nie ma, komputer prosi o jego wprowadzenie. Zrobi tak tylko pierwszy raz, potem będzie już od razu uruchamiał podprogram przeglądania danych.
W opcji przeglądania zapisów można je przeglądać w/g ich kolejności zapisu w pamięci lub zawartości kluczy, kasować wybrane zapisy lub wprowadzać nowe. Podczas

```
BAZA DANYCH 64
(C) 1992 KEBAB

MENU
1 - ZAKLADANIE BAZY
2 - ODCZYT BAZY (DYSK)
3 - ZAPIS BAZY (DYSK)
4 - OPERACJE NA ZAPISANYCH DANYCH
5 - SORTOWANIE DANYCH W/G KLUCZA

NAZWA BAZY:
ILOSC DOKONANYCH ZAPISOW: 0

WYBIERZ OPCJE KLAWISZEM OD 1 DO 5
```

wyszukiwania zapisu możemy się posługiwać zawartością pól-kluczy, np. gdy chcemy przejrzeć wszystkie książki konkretnego autora. Gdy nie pamiętamy dokładnie jego nazwiska, wystarczy wprowadzić tylko jego początkowe litery, a te, których nie jesteśmy pewni, należy zastąpić znakiem „?". Po wprowadzeniu szukanego nazwiska, komputer przeszuka wszystkie zapisy w poszukiwaniu książek napisanych przez tego autora. Po znalezieniu każdego z poszukiwanych zapisów następuje jego wydrukowanie na ekranie i pytanie o pozwolenie kontynuacji szukania.
Zdefiniowaną bazę danych możemy zapisać na taśmę lub dyskietkę - urządzenie można każdorazowo wybrać przed zapisem. To samo dotyczy się operacji odczytu. Przy starcie programu typ urządzenia jest ustawiany zgodnie z urządzeniem ostatnio przez użytkownika używanym.
Ostatnią opcją jest sortowanie. Polega ono na przestawieniu zapisów w pamięci według określonej kolejności (alfabetycznej lub rosnącej). Po wybraniu tej opcji komputer drukuje nam na ekranie wszystkie nazwy kluczy i prosi o wskazanie tego, w/g którego odbyć się ma sortowanie. Po wykonaniu obliczeń wszystkie zapisy znajdują się już w nowej kolejności.
Nasza mała baza danych może pomieścić maksymalnie 300 zapisów gdzie każdemu z nich można przydzielić maksymalnie do dziesięciu kluczy. Cały program sprawdza się u mnie dobrze jako podręczna książka telefoniczna.

Paweł „Polonus" Sołtysiński



Listing nr 1 na str. 30



Czy wiedzieliście, że...

...zmiana zawartości jednej komórki pamięci wystarczy, aby unieszkodliwić moduł Action Replay MK II? Jeżeli w momencie wciśnięcia przycisku FREEZE pod adresem zawierającym młodsze słowo adresu copperlisty ($dff082 lub $dff086) znajduje się liczba nieparzysta, to po chwili mrugania diodą komputer wyświetli okienko GURU z numerem błędu równym 287. Warto wspomnieć, że jeśli uprzednio „namieszaliśmy" w bibliotekach systemowych to restart systemu dokona się z całkowitym czyszczeniem pamięci. I co Wy na to, rycerze spod znaku Action Replay'a?

...copperlista umieszczona od adresu $4c0 pojawiać się będzie na kilka dziesiątych sekundy po każdorazowym wciśnięciu kombinacji Control-Amiga-Amiga? Jako, że wszystkie wektory systemowe związane z procedurą resetowania pozostają niezmienione, wszelakiej maści virus-killery nie mają o owym fakcie zielonego pojęcia i zachowują się w sposób bierny! Nie należy jednak obawiać się inwazji „fagów" wykorzystujących tę mało znaną właściwość systemu operacyjnego Amigi. Co prawda zręcznie napisana copperlista jest w stanie uruchomić wirusa (lub dowolny inny program), lecz wymagane jest do tego ustawienie bitu CopperDangerBit, który na szczęście w chwili restartu systemu jest wyzerowany. I jeszcze jedno - adres $4c0 sprawdza się tylko w przypadku komputerów wyposażonych w pamięć SLOW i/lub FAST, w pozostałych (posiadających tylko CHIP) Amigach lokacja programu dla Copper'a musi być inna, jakkolwiek nie jesteśmy na razie w stanie podać Wam jaka.

Miłosław „Thorgal" Smyk

Masterseka-HELP

Na prośbę wielu Czytelników, prezentujemy poniżej komendy edycyjne i dyrektywy (przetłumaczony zbiór "HELP") asemblera „MasterSeka v1.71".

Polecenia występujące w pliku 'S:MasterSeka.startup':
-a adres : Alokacja obszaru roboczego od podanego adresu absolutnego.
-b znacznik : ScrollBar 0-brak, 1-z prawej, 2-z lewej.
-c : Alokacja obszaru roboczego w Chip-Ram.
-e ilość : Ilość linii edytora.
-h ilość : Ilość użytych poleceń debuger'a, dostępnych za pomocą klawiszy kursor-góra, kursor-dół.
-l znacznik : Numeracja lini 0-wyłączona, 1-włączona.
-r znacznik : Rodzaj użytego requestera 0-żaden, 1-arp.library, 2-req.library.
-s ilość : Ilość bit-plane'ów dla ekranu Seki, jeśli 0 to otworzy okno.

Polecenia debugger'a (dane opcjonalne w nawiasach kwadratowych):
= : Informacja o wielkości source'a, objektu, tablicy relokacji, danych.
? [wyr. arytm.] : Podręczny kalkulator.
a : Asemblacja.
b : Przejście do dołu source'a w edytorze.
b Adres : Obliczenie sumy kontrolnej dla BootBlock'u.
c : Porównanie obszarów pamięci.
cl : Kasowanie source'a z edytora.
cls : Czyszczenie ekranu.
cs : Tworzenie tablicy sinusów (niezbędna biblioteka mathtrans.library).
d [adres] : Disasemblacja.
f : Przeszukiwanie pamięci.
fi : Wypełnianie pamięci.
g [adres] : skok pod adres (JMP).
h : Patrz „=".
j [adres] : skok pod adres (JSR).
kf : Kasowanie pliku z urządzenia zewnętrznego.
ks : Kasowanie source'a z edytora.
m [adres] : Modyfikacja pamięci.
n [adres] : Wyświetlanie znaków ASCII.
o : Odzyskanie source'a po „ks".
q [adres] : Wyświetlanie zawartości pamięci.
r [nazwa pliku] : Odczyt source'a.
ri [nazwa pliku] : Odczyt pliku.
ro [nazwa pliku] : Odczyt objektu, czyli segmentu.
rs [numer napędu] : Odczyt sektorów.
rt [numer napędu] : Odczyt track'ów.
s [adres] : Tryb „trace" - śledzenie wykonywania programu.
t [numer linii] : Przejście do góry source'a w edytorze lub skok do podanej linii.
v [ścieżka] : Odczyt katalogu.
v ścieżka : Zmiana katalogu.
w [nazwa pliku] : Zapis source'a.
wb znacznik : 0 - zamknięcie workbench'a, 1 - otworzenie workbench'a.



Arexx...

... część druga

Serdecznie zapraszamy do lektury rozpoczętego przed miesiącem krótkiego przewodnika po ARexx'ie.

Trochę komplikacji.

Czy kiedykolwiek zdarzyło się Wam podczas pisania programu, że potrzebowaliście tablicy kodów ASCII? Idę o zakład, że tak. Czemu więc nie ułatwić sobie życia i nie napisać stosownej procedury w ARexx'ie? Taki program przedstawia listing 2. Mamy w nim coś nowego: pętlę i instrukcję warunkową. Zastosowana pętla jest jedną z najprostszych oferowanych przez Rexx'a (w sumie mamy chyba 5 typów pętli). Zaczyna się słowem kluczowym „DO". Pozostała część instrukcji jest taka sama jak dla FOR w Basicu - mamy zmienną sterującą „i", ograniczenie dolne i górne. Gdybyśmy chcieli określić skok zmiennej sterującej (domyślny to 1), powinniśmy to zrobić tak:

```
DO i=32 TO 127 BY 2
```

aby przebiec wartości od 32 do 127 co 2. Pętla kończy się instrukcją „END". Nowa dla Was jest jeszcze instrukcja warunkowa. Nie różni się ona od tych używanych w Basicu czy Pascalu i wygląda następująco:

```
IF <WarunekLogiczny> THEN
   <instrukcja>
ELSE
   <instrukcja>
```

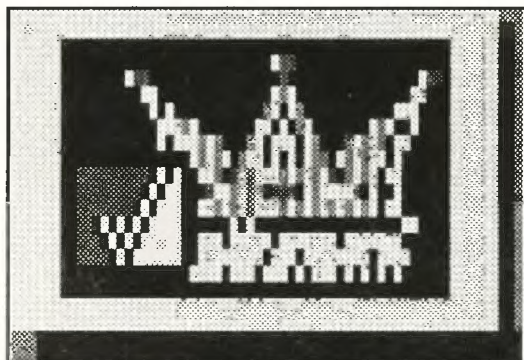
Jeżeli <WarunekLogiczny> jest spełniony, to zostanie wykonana instrukcja po THEN, w przeciwnym wypadku - po ELSE. Należy zauważyć, że zarówno po THEN, jak i ELSE może być tylko jedna instrukcja. Może to być jednak tzw. instrukcja złożona składająca się z dowolnej ilości instrukcji prostych. Tworzymy ją przez umieszczenie żądanych instrukcji pomiędzy słowami kluczowymi DO i END, jak w przykładzie:

```
IF 1=1 THEN
DO
   a='True'
   Okay1 a
END
ELSE
DO
   a='False'
   Okay1 a
END
```

W naszym programie pojawia się jeszcze jedna nowa funkcja:
D2C(numer)
Zwraca ona znak o kodzie ASCII równym wartości <numer>. Myślę, że po tych wyjaśnieniach sposób działania programu jest już jasny. Niestety, pierwsza próba uruchomienia procedury przynosi rozczarowanie - czemu tak wolno? Tutaj ujawnia się największa wada ARex-x'a - powolność działania.
Należy w tym miejscu zaznaczyć, że wszystkie wbudowane funkcje są dosyć szybkie, ale przetwarzanie pętli, instrukcji warunkowych wprowadza znaczące opóźnienia. No cóż, może w przyszłych wersjach ARexx'a (ten artykuł oparty jest o wersję 1.15) zostanie to poprawione. Ale... pomyślmy. Nasz program za każdym razem tworzy tabelę na nowo. A przecież wystarczyłoby ją zrobić raz, przechować gdzieś w pamięci i przy każdym kolejnym uruchomieniu tylko odczytywać. Czy jest to jednak możliwe? Oczywiście, wystarczy spojrzeć na listing 3.
Jak widać w tym przypadku odwołujemy się do tzw. „Clipboard'u". Clipboard to po prostu coś w rodzaju zmiennej, która pozostaje w pamięci nawet po zakończeniu wykonywania programu w przeciwieństwie do normalnych zmiennych, które po wykonaniu programu ulegają skasowaniu. Do obsługi clipboard'ów używamy dwóch funkcji:
GETCLIP(NazwaClipboard'u)
zwraca ona zawartość danego clipboard'u, lub ciąg pusty w przypadku, gdy clipboard nie istnieje.
SETCLIP(NazwaClipboard'u,Tekst)
zapisuje do odpowiedniego clipboard'u podany przez nas tekst.
Warto zauważyć, w jaki sposób wywoływana jest ta druga funkcja. Mianowicie:
CALL Funkcja(parametry).
Takie odwołanie „mówi" ARexx'owi, że nie interesuje nas wynik działania funkcji i może zostać pominięty.

Trochę czarnej magii.

Zadanie, jakie teraz przed nami stoi, to zapewnienie użytkownikowi Cygnusa możliwości obliczania wyrażeń arytmetycznych. W tradycyjnych językach (Pascal, C) odbywa się to poprzez mozolne analizowanie ciągu tekstowego. ARexx robi to jedną instrukcją. Na początku programu (listing 4) występują dwie nowe instrukcje.
OPTIONS RESULTS deklaruje, że program użytkowy (w tym wypadku CED), z którym zamierzamy się



komunikować, będzie nam coś przekazywał.
SIGNAL ON SYNTAX powoduje natomiast, że w przypadku napotkania błędu składniowego nie wystąpi przerwanie wykonywania programu, ale skok do etykiety SYNTAX. Etykieta ta znajduje się na końcu programu.
GETSTRING (instrukcja Cygnusa) powoduje wyświetlenie na ekranie Cygnusa okienka (podobnego do tego, jakie pojawia się np. przy wybraniu opcji „Search"), do którego możemy wpisać tekst. Wprowadzony przez nas ciąg jesteśmy w stanie odczytać ze zmiennej ARex-x'a o nazwie RESULT. Na podobnej zasadzie działają instrukcje Cygnusa GETNUMBER i GETFILENAME - służą one odpowiednio do pobierania liczb lub wybierania nazw plików. Podane przez użytkownika wyrażenie natychmiast musimy zapamiętać w jakiejś zmiennej, bowiem zawartość RESULT bardzo często się zmienia. Musimy następnie sprawdzić, czy użytkownik rzeczywiście nam coś przekazał. Jeżeli bowiem wybrano „Cancel", to zmienna RESULT przyjmie wartość „RESULT" (sic!), a gdy naciśnięto tylko RETURN - zawierać będzie ciąg pusty. W tym przypadku stosujemy operator logiczny AND (znaczek &) i NOT (znaczek ~). Należy zauważyć, że nie możemy napisać w ARexx'ie na przykład tak:

```
IF 1<>2 THEN ...
```
a jedynie
```
IF 1~=2 THEN ...
```

chociaż oba te zapisy znaczą dokładnie to samo, tyle że w innych językach. Inne dozwolone operatory logiczne to OR (znaczek |) i XOR (znaczek ^). Jeżeli użytkownik coś wprowadził, to tekst ten składamy z ciągiem „a=". Na przykład jeżeli wpisane zostało „23+45*2", to w zmiennej Oblicz znajdzie się tekst: a=23+45*2
który bądź co bądź przypomina wyrażenie z Rexx'a... Jeżeli zaś przypomina - czemu by go nie obliczy? Dokonujemy tego instrukcją:

```
INTERPRET <tekst>
```

traktującą <tekst> jako fragment programu w Rexx'ie i interpretującą go. W przypadku wprowadzenia przez użytkownika sensownego wyrażenia, zostanie ono obliczone, zaś otrzymana wartość będzie przypisana zmiennej <a>, która następnie zostanie wydrukowana. Gdy natomiast wyrażenie nie jest poprawne, to nastąpi skok do etykiety SYNTAX i użytkownik będzie powiadomiony o fakcie popełnienia błędu. Nowa dla Was instrukcja:

```
EXIT <KodPowrotu>
```

powoduje zakończenie wykonywania programu i zwraca programowi wywołującemu kod błędu (0 oznacza poprawne zakoczenie; teoretycznie więc powinniśmy ustawić kod błędu na np. 5, co oznaczałoby błąd, ale to w naszym przypadku niesie wyłącznie niedogodności).
Myślę, że warto na marginesie wspomnieć, jakiego typu liczby i operatory arytmetyczne akceptuje ARexx. Jeżeli chodzi o liczby, to dopuszczalne są wszystkie - zmiennoprzecinkowe lub stałoprzecinkowe - wartości w granicach od -1E308 do 1E308 (tak w przybliżeniu). Prócz 4 typowych operacji arytmetycznych (+ - * /) i nawiasów mamy też:
* potęgowanie, oznaczane dwoma gwiazdkami (np. 2 do potęgi trzeciej zapisujemy jako 2**3);
* resztę z dzielenia (wspomniana już wyżej operacja „modulo"), czyli podwójna ukośna kreska (np. resztę z dzielenia 40 przez 3 wyrazimy jako 40//3 i jest ona równa 1);
* dzielenie całkowite, oznaczane przez znak procent (np. dzielenie całkowite 23 przez 7 zapiszemy przez 23%7). Nie ma natomiast funkcji pierwastka, logarytmów czy funkcji trygonometrycznych (choć są one dostępne w specjalnej bibliotece).

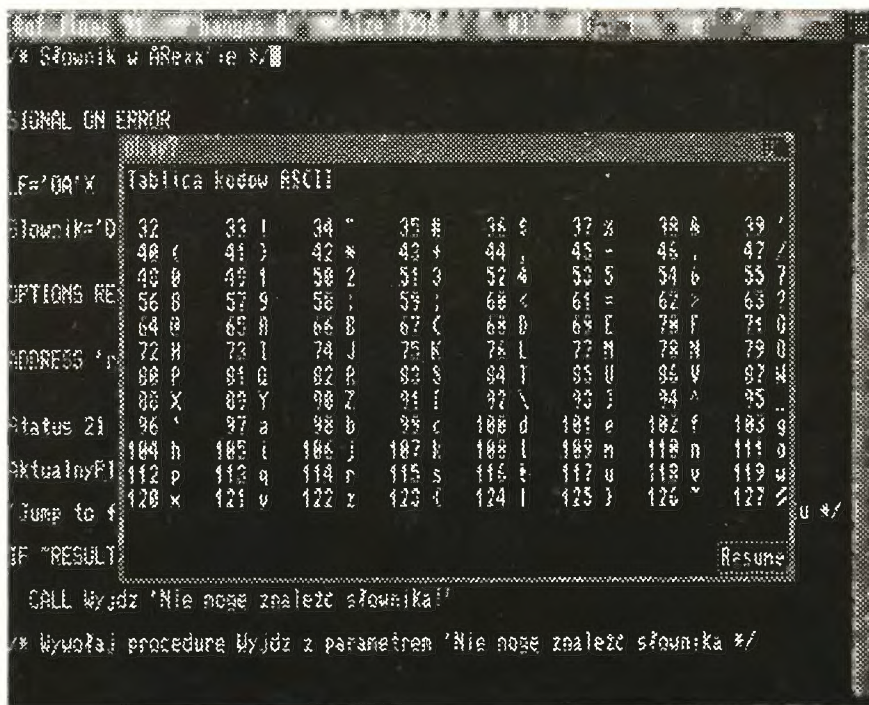
Last but not least.
Ostatni (już słyszę ten jęk zawodu!) prezentowany w tym miejscu program będzie prawdziwie profesjonalny. Za podobną „twórczość" niektórzy ludzie biorą grube pieniądze - a tu dostajecie go za darmo No, ale do rzeczy. Zamierzam przedstawić Wam prosty (co wcale nie znaczy, że prymitywny!) słownik. Mógł on powstać dzięki wspaniałym możliwościom Cygnusa. Zauważcie bowiem, że edytor ten przeprowadza wyszukiwanie tekstów z piorunującą wręcz szybkością. Druga jego istotna cecha to możliwość jednoczesnej edycji do dziesięciu tekstów. Mam nadzieję, iż teraz zrozumieliście już mój zamysł - po prostu jednym z tekstów Cygnusa będzie nasz słownik, w którym przy użyciu opcji „Search" będziemy dokonywać wyszukiwania. Teraz wystarczy już tylko napisać stosowny program w ARex-x'ie. Ponieważ zawiera on już wcześniej poznane elementy języka, nie zamierzam go omawiać - zwłaszcza, że posiada komentarze. Warto natomiast poświęcić kilka słów jego obsłudze.





Pierwsze, co musimy zrobić, to wczytać słownik do Cygnusa (przez użycie opcji „Open new" i „Open..."), przy czym powinien on nosić nazwę „Dictionary.txt" (drobna modyfikacja w programie pozwala na zmianę tego warunku). Jeżeli nie mamy słownika, to musimy go założyć - otwieramy nowy tekst, naciskamy RETURN dla wprowadzenia pustej linii (ważne!) i nagrywamy plik jako „Dictionary.txt". W celu wyszukania jakiegoś słowa naciskamy odpowiedni klawisz funkcyjny i... na ekranie pojawia się okienko, gdzie wprowadzamy poszukiwany wyraz. Jeżeli zostanie on znaleziony, to program wyświetli jego definicję; w przeciwnym wypadku - będziemy poproszeni o podanie opisu i słownik zostanie uzupełniony o nowe słowo. Możemy zresztą na własną rękę wypełnić słownik. Należy jednak trzymać się następujących zasad:
- słowo i jego definicja muszą mieścić się w jednej linii. W przypadku wprowadzania długich określeń możemy odpowiednio zmienić po-

łożenie prawego marginesu);
- format wprowadzanych definicji powinien być następujący: <słowo><spacja><myślnik><spacja><opis słowa>
Na przykład:
home - dom, mieszkanie
computer - komputer
Do słownika można wprowadzić wiele ulepszeń, przykładowo stosując kompresję opisów.



Zakończenie.
W swoim artykule starałem się pokazać zastosowanie Amigi od strony praktycznej. ARexx bardzo w tym pomaga - i jest to chyba jedna z przyczyn, dla których Commodore tak gorąco go popiera. Możliwe, że już za rok istnienie w programie interface'u dla ARexx'a będzie koniecznością i np. będę Wam mógł zaprezentować kilka makro-procedur dla Deluxe Paint'a 5... Tymczasem jednak, ze względu na brak na naszym rynku książek o ARexx'ie, jedynym sposobem nauczenia się programowania w tym języku jest próbowanie, próbowanie i jeszcze raz próbowanie. Wielu nocy spędzonych przy klawiaturze życzy:

Michał Łętowski

Listingi nr 2,3,4,5 na str. 32

64

Piszemy po polsku !

polskie litery dla C-64

Problem polskich znaków we własnych programach pisanych w BASIC (i nie tylko) potrafi być uciążliwy dla początkujących programistów. To nie to samo co poczciwy ZX Spectrum, tu nie definiuje się nowych znaków przy pomocy jednego rozkazu BASIC. Instalowanie własnego liternictwa jest czasami bardzo kłopotliwe nawet dla tych, którzy znają zasady ale robią to po raz pierwszy.
Aby więc maksymalnie to ułatwić, przygotowaliśmy cały zestaw znaków wraz z polskimi znakami specjalnymi. Program „Polskie Znaki 64" należy wpisać do pamięci komputera przy pomocy KOREKTORA lub przy użyciu dowolnego monitora z tym, że pomijamy wtedy kody kontrolne, umieszczone w nawiasach. Przy okazji przenieśliśmy ekran tekstowy z obszaru $0400-$07e8 do $c400-$c7e8. Taka zmiana umożliwiła przesunięcie początku pamięci dla programów w języku BASIC o 1024 bajty w dół, co daje nam 1 kilobajt więcej dla realizacji naszych planów.
Po zainstalowaniu polskich znaków możemy wgrywać do pamięci wszystkie posiadane programy w BASIC, pod warunkiem, że nie odwołują się do pamięci ekranu za pomocą instrukcji POKE (jak np. zamieszczona w tym numerze gra).

Paweł Sołtysiński

Listing nr 6 na str. 33



PC-Task

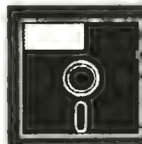
czyli udawania „Big Blue" ciąg dalszy...

W poprzednim numerze Kebaba przedstawiliśmy dwa sprzętowe emulatory komputera IBM PC dla Amigi. Mają one swe plusy i minusy w dziedzinie szybkości i kompatybilności. jednakże dla przeciętnego użytkownika najbardziej znacząca i jednocześnie zniechęcająca do kupna będzie cena owych przystawek, nierzadko przekraczająca połowę wartości nowej Amigi 500. Wady tej nie posiadają rozwiązania programowe, co prawda sporo wolniejsze, ale jeśli idzie o zgodność z oryginałem nie pozostające w tyle za swymi wyekwipowanymi w hardware kuzynami. Jednym z nowszych takich emulatorów jest PC-Task autorstwa Chrisa Hamesa z Australi.

PC-Task v1.04 umożliwia nam pracę w trybie MDA lub CGA czym już na starcie bije sporo starszego Transformera, który jak pamiętamy nie posiadał trybu graficznego ani kolorów. Wyboru rodzaju karty graficznej dokonujemy przy uruchamianiu programu korzystając z estetycznie zaprojektowanego panelu, pozwalającego nam także na ustalenie ilości pamięci jaka będzie do dyspozycji MS-DOS'u, nazwy urządzeń mających stać się odpowiednikami IBM'owskich stacji A: i B: oraz twardego dysku C: i D: (nic zatem nie stoi na przeszkodzie posiadaczom zasobnych w pamięć operacyjną komputerów w umieszczeniu „twardziela" na RAM-dysku, która to podmiana znacznie przyspiesza pracę w przypadku programów często odwołujących się do plików tymczasowych), typu portów jakie emulator będzie obsługiwał i, uwaga, priorytetu z jakim uruchomi się PC-Task. Tak, to nie pomyłka, program pracuje w multitaskingu, zapewniając jednoczesny dostęp do aplikacji MS- i AmigaDOS'u. Co prawda oprócz oczywistych korzyści niesie to za sobą także pewne niedogodności, a mianowicie podobnie jak w przypadku sprzętowego ATonce opóźnienia w dostępie do dysku i nieznaczne zmniejszenie prędkości działania. Są one jednakże pomijalne, szczególnie dla osób przyzwyczajonych do pracy z Amigą (użytkowników PC może początkowo konfudować konieczność odczekania kilku sekund przed uzyskaniem dostępu do właśnie włożonej dyskietki). Z kolei już podczas emulacji mamy możliwość uruchomienia handlera myszy, dzięki któremu nasz amigowy gryzoń może współpracować z takimi programami jak na przykład Norton Commander czy Turbo Pascal 6.0.



Tempo z jakim PC-Task „przedziera" się przez kod procesora 8086 nie jest być może oszałamiające, należy jednak pamiętać, że każdy dodatkowy megaherc w zegarze naszej Amigi powoduje również wzrost szybkości pracy programu, co niestety nie zawsze da się powiedzieć o emulatorach sprzętowych. Testy wykazały, iż przy aktywnej emulacji MDA, PC-Task jest sporo szybszy od Transformera - nawet do dwóch razy - szczególnie gdy wykonywanych jest wiele operacji na ekranie. Prędkość obliczeń oszacowaliśmy nieco inaczej niż zwykle, zamiast bowiem skorzystać z System Info Nortona, zmierzyliśmy czas kompilacji tysiąca linii typowego programu z wykorzystaniem Turbo Pascala 5.5. Rezulaty są następujące: Transformer v1.21 - 1 min. 48 sek., PC-Task (MDA) - 1 min. 19 sek., PC-Task (CGA) - 2 min. 50 sek. Ten ostatni rezultat byłby z pewnością lepszy na Amidze wyposażonej w prawdziwy FAST RAM, jako że poprawne przedstawienie szesnastokolorowego trybu tekstowego CGA wy-



```
PC-Task V1.04. Copyright 1992 Chris Hames. All rights reserved.

To start the mouse driver press LeftAmiga-P
To quit press the left/right amiga key's and Del.

Memory Available: 640K
Drive A is: DF0
Drive B is: Unavailable
HardDrive C is: DH1:emulators/smutniak/PCTask/hardfile0
HardDrive D is: Unavailable

C>echo off

MS-DOS Version 5.00

C:\>
```

PC-Task w akcji...!

maga użycia czterech bitplane'ów o wysokiej rozdzielczości, a to pociąga za sobą znaczne spowolnienie wykonywania programu umieszczonego w pamięci CHIP lub SLOW.
Trzeba przyznać, że na zwykłej, nieprzyspieszonej Amidze 500 czy 2000 żółwie tempo pracy PC-Task-'a niezwykle utrudnia wykorzystywanie go do jakichkolwiek poważniejszych zadań. Warto jednak pamiętać iż jeszcze kilka lat temu, w dobie komputerów XT, tempo to należało do przeciętnych i zupełnie wystarczało do tworzenia złożonych aplikacji w dajmy na to, Turbo Pascalu. Tak więc emulator można polecić osobom posiadającym sporą dozę cierpliwości, tudzież szczęśliwym właścicielom Amig 3000 lub tzw. „dopalaczy".
I jeszcze kilka słów do Czytelników zainteresowanych kupnem opisywanego programu. Jest on do nabycia drogą wysyłkową u autora (adres zamieszczamy poniżej) w cenie 35 dolarów amerykańskich. Za sumę tę otrzymujemy PC-Task'a v1.04 w wersjach dla procesora 68000, 68010 oraz 68020 i kolejnych, pełną instrukcję obsługi w języku angielskim, dwa programiki dla MS-DOSu pozwalające przesyłać dane na linii IBM - Amiga poprzez dyskietki oraz, w wypadku problemów, możliwość darmowego korzystania z porad twórcy programu.

Chris Hames
6 Pamela Crt
Blackburn Sth
VIC 3130
Australia

Miłosław „Thorgal" Smyk

64

Kupiłem C-64...

i co dalej? (cz.5)

Witam wszystkich w kolejnym artykule dla „świeżych" posiadaczy Commodore 64. Zostało nam jeszcze kilka rzeczy do wyjaśnienia, które się Wam na pewno przydadzą w programowaniu w języku BASIC. Przede wszystkim zostały nam do omówienia tzw. zmienne tablicowe. Co to jest? Jeżeli chodzi o ich rodzaje - to bez większych zmian, tzn. mamy tutaj te same rodzaje zmiennych jak w przypadku zmiennych zwykłych, tzn. tekstowe (ze znakiem $), liczbowe i liczbowe całkowite (ze znakiem %). No to gdzie te różnice, spytacie? Otóż, zmienne tablicowe pozwalają na zapisanie więcej niż jednej danej w jednej zmiennej. Nie można na przykład przypisać zmiennej A jednocześnie wartości 10,30 i 50, gdyż jest to po prostu niemożliwe. Aby tego typu operacje mogły mieć miejsce, zmienne takie postanowiono zapisywać z dodatkowym parametrem, który wskazuje interesujące nas dane, np.:

```
10 DIM A(3)
20 A(1)=10:A(2)=30:A(3)=50
30 PRINT A(1);A(2);A(3)
```

Jak widać, zmienna tablicowa to taka zmienna, po której nazwie umieszcza się w nawiasach wspomniany wyżej parametr. Zapisanie i odwoływanie się do takiej zmiennej zawsze musi się odbywać ze wskazaniem, do którego pola w zmiennej operacja ta ma się odnosić. Zwróćmy teraz uwagę na linię nr 10 w naszym krótkim przykładzie. Zawarty jest w niej bowiem specjalny rozkaz j. BASIC, który przygotowuje zmienną tablicową do pracy. Po słowie DIM wystarczy napisać nazwę zmiennej a w nawiasie jej rozmiar. Przygotować zmienną tablicową należy przed jej użyciem, najlepiej zaraz na początku programu lub po każdorazowym użyciu rozkazu CLR (kasowanie wszystkich zmiennych).
Specjalne rodzaje zmiennych (tekstowe i liczbowe całkowite) uzyskuje się przez dodanie po ich nazwie parametru w nawiasach, np. A$(10) lub B%(2).
O ile zasady posługiwania się tablicami jednoparametrowymi są względnie jasne, to dwu- i trójparametrowe mogą być nieco bardziej kłopotliwe. O ile jednak graliście kiedyś na kartce papieru w kratkę w „bitwę morską", to bez trudu „złapiecie" zasadę. Dwa parametry w zmiennej tablicowej (np. A$(10,30) opisują położenie danych w tablicy prostokątnej (a w szczególnym jej przypadku - kwadratowej) na zasadzie układu współrzędnych. Definiowanie takich zmiennych powinno wyglądać w ten sposób: np. DIM A$(20,30). O ile wyznaczoną tymi wymiarami tablicę (20 x 30) możemy sobie wyobrazić jako twór płaski to poprzez dodanie do definicji zmiennej tablicowej trzeciego parametru możemy stworzyć kilka takich prostokątnych płaszczyzn, dostępnych w jednej zmiennej. Definiujemy je analogicznie - np. DIM A$(10,10,10). Tablicę taką możemy nazwać trójwymiarową.
Do czego najczęściej stosuje się takiego typu zmienne? Wszędzie,

gdzie napisany przez nas program musi gromadzić większe ilości danych i je obrabiać. Rozwiązanie takie daje dużo większą prędkość i przejrzystość programu, nie mówiąc już o jego długości. Przykład wykorzystania tych zmiennych możemy znaleźć w zamieszczonym w tym numerze programie typu baza danych.
W definicji języka BASIC znajduje się jeszcze kilka rozwiązań, które mają na celu zmniejszanie objętości programu i zwiększenie jego czytelności dla późniejszego użytkownika. Jednymi z najdogodniejszych są instrukcje pętli FOR/NEXT. Jak to działa? Obejrzyjmy przykład:

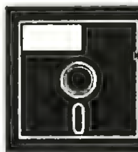
```
10 FOR T=1 TO 5
20 PRINT T;
30 NEXT T
RUN
 1 2 3 4 5
```

A teraz opis działania tej pętli: instrukcja FOR oznacza przypisanie zmiennej T (która od tej chwili będzie licznikiem pętli) wartości startowej 1. Potem następuje zdefiniowanie wartości końcowej poprzez parametr poprzedzony słowem TO. Oznacza on wartość, po osiągnieciu której następuje wyjście z pętli. Instrukcja NEXT T (gdzie T to nazwa zmiennej będącej licznikiem pętli) powoduje zwiekszenie T o jeden i porównanie owej zwiększonej wartości z parametrem 5 w linii 10. Jeżeli jest równy, to następuje ostatnie wykonanie tej pętli, po czym program kontynuowany jest od miejsca za komendą NEXT (w naszym przykładzie nic już dalej nie ma, więc program się kończy w tym właśnie miejscu). Podczas wykonywania pętli powtarzany jest jest fragment programu pomiędzy definicją pętli (FOR T=1 TO 5) a jej końcem (NEXT T).
Jak nietrudno zauważyć, zmienna-licznik pętli jest każdorazowo zwiększana o jeden. Co jednak wtedy, gdy chcemy, by była ona zwiekszana o np. dwa lub nawet -2? Do tego celu służy jeszcze jedna instrukcja - STEP (po polsku - krok), która reguluje wielkość, o jaką zwiększana jest zmienna-licznik (w naszym przykładnie T). Przeanalizujmy ten przykład:



```
10 FOR A=4 TO 8
STEP 2
20 PRINT A;
30 NEXT A
RUN
 4  6  8
```

Dla dociekliwych propozycja - spróbujcie wstawić po STEP np. 0.5 zamiast 2.
Na koniec jeden ważny szczegół - jako licznik może służyć jedynie zmienna liczbowa. Oraz jeden mniej ważny szczegół - po rozkazie NEXT możemy opuścić nazwę zmiennej.



Paweł Sołtysiński

64

Żarłoczny robaczek !

prosta gra dla C-64

W tym numerze KEBABA sporo dla siebie mogą znaleźć miłośnicy języka BASIC. Dla nich dedykowany jest ten program - prosta gra zręcznościowa. Należy go starannie przepisać i nagrać na taśmę lub dyskietkę przy pomocy komendy SAVE. Ponieważ w aktualnym odcinku poradnika „Kupiłem C-64 i co dalej?" omawiane były między innymi zmienne tablicowe, zdecydowałem się napisać prostą grę, gdzie użycie tablic zapewniałoby należytą prędkość programu.
A o co chodzi w samej grze? Po jej uruchomieniu należy poczekać kilkadziesiąt sekund, aż komputer przygotuje odpowiednie dane dla zmiennej tablicowej. Rozpoczęcie gry odbywa się później poprzez wciśnięcie przycisku FIRE na joystick'u podłączonym do portu drugiego. Joystick'iem sterujemy ruchami naszego bohatera - żarłocznego robaczka. Robaczki zwykle zajmują się jedzeniem więc i nasz nie jest tu wyjątkiem. Jego ulubionym pokarmem są okrągłe czekoladki (O). Jako pokarm muszą być one bardzo wydajne, bowiem zjedzenie każdej z nich powoduje przyrost naszego robaczka o jeden element. Sterowanie owadem staje się coraz trudniejsze, zważywszy iż nadgryzienie otaczającego pole gry murku, jego pojawiających się elementów oraz uszkodzenie samego siebie kończy grę! Pomimo swojej prostoty, gra wydaje się być wciągająca (mój rekord to 83 punkty!) oraz da na pewno wiele materiału do przemyśleń dla początkujących programistów.

Miłej zabawy życzy wszystkim Czytelnikom

Paweł Sołtysiński

Listing nr 7 na str. 34



Digi Tiger II

najszybszy z powolnych

Niektórzy z naszych Czytelników pamiętają jeszcze z pewnością czasy pierwszej fascynacji Amigą i jej możliwościami graficznymi. Na rok 1985 były one faktycznie niesamowite. Przypomnijmy sobie, że wtedy C-64 był jednym z najlepszych komputerów w tej dziedzinie (mimo tego, że nie znano jeszcze rozmaitych kruczków programowych umożliwiających uzyskanie jeszcze lepszych rezultatów). W królestwie „poważnych informatyków" występował IBM PC ze swoją świtą „sępów" tj. rozmaitych firm i firemek konstruujących (bądź tylko kopiujących) komputery na wzór wielkiego „Big Blue". Standardem grafiki w tym królestwie były dwie karty o dumnie brzmiących nazwach - „Hercules Graphics Card" oraz „Color Graphics Adapter". Pierwszą z nich należy chyba (mimo kilku wad typu brak koloru czy niestandardowe proporcje wymiarów punktów) uznać za jedną z najbardziej udanych (co potwierdziło się w praktyce) konstrukcji dla zastosowań biurowych. Efekty uzyskiwane przy użyciu tej drugiej (CGA) nie dorównywały niestety tym, które można uzyskać na C-64. Ale to już na szczęście (dla miłośników standardu PC) historia. W tą właśnie historię wkroczyła Amiga. 4096 kolorów jednocześnie na ekranie, 16 kolorów jednocześnie przy rozdzielczości 640x400 (512) punktów (nie znano jeszcze odpowiedników efektów typu FLI na C-64 które umożliwiają uzyskanie całej palety kolorów na ekranie w najwyższej rozdzielczości - Dynamic Hires) to możliwości, które zamknęły na dłuższy czas usta konkurencji a prasa fachowa określała je przymiotnikami typu „amazing", „fantastic" czy „photo-realistic". Oczywiście nie po polsku bo nasi rodzimi „specjaliści" byli wówczas na etapie zachwalania wciskanego klientom przez monopolistyczny jeszcze „Pewex" komputera Atari 800. To na szczęście (dla nas - Commodore'owców) również już tylko historia. Jedno z pierwszych zastosowań jakie się nasunęło na myśl tym, którzy postanowili sprawdzić co właściwie warta jest ta cała Amiga to... cyfrowe przetwarzanie obrazu. Przy okazji pojawiły się też pierwsze programy demonstracyjne wykorzystujące animację digitalizowanych (jeżeli jesteś miłośnikiem czysto polskich określeń - czytaj: przetworzonych na postać cyfrową) obrazów. Aby taki obraz uzyskać potrzebne jest tzw. digitizer czyli specjalne urządzenie, które dokona wyżej wspomnianego przetworzenia. Różne są metody uzyskiwania cyfrowej postaci obrazu (na przykład ręczny skaner to również swego rodzaju digitizer). Nas jednak dzisiaj interesować będą tylko digitizery umożliwiające przetworzenie obrazu zakodowanego w tzw. standardzie Composite Video (patrz słowniczek). Dzisiaj, w czasach gdy w bardzo wielu domach znajdują się magnetowidy, a w niektórych również amatorskie kamery video, nietrudno o uzyskanie takiego sygnału. Większość kamer oraz magnetowidów domowych posiada wyjścia sygnałowe pracujące w tym właśnie standardzie. Jeżeli mamy już źródło sygnału tzn. kamerę lub magnetowid to szybko kupujemy sobie video digitizer... tak... tu zaczynają się pierwsze kłopoty. Rozpiętość cenowa jest spora. Od około 2 mln zł. do... strach pisać ile! Aby uchronić się (i Czytelników) od zbędnych szoków nerwowych, założymy sobie następujące punkty, którymi będziemy się kierować przy wyborze sprzętu. Po pierwsze nasz digitizer powinien przetwa-



Amiga, kamera, Digi Tiger...



Tak przedstawia się nasz mały tygrys...

rzać obrazy zarówno czarno-białe jak i kolorowe. Po drugie cena nie powinna przekraczać granic przyzwoitości. No i wybór się upraszcza. W zasadzie mamy tylko cztery możliwości: Digi-View Gold 4.0 + RGB Splitter, Deluxe View + RGB Splitter, Deluxe View wersja Proline One oraz... No właśnie! Digi Tiger II. Cena wszystkich tych zestawów jest bardzo zbliżona. Różnice występują w szybkości działania oraz w jakości przetwarzania obrazu. Ze względu na bardzo wysokie notowania osiągnięte w testach niezależnych czasopism zachodnich postanowiliśmy „wziąć na warsztat" ostatnie z wyżej wymienionych urządzeń. Digi Tiger II to (podobnie jak Deluxe View wersja Proline One) zintegrowany w jednej obudowie video digitizer oraz RGB Splitter (patrz słowniczek). Konstrukcja tego typu umożliwia w pełni automatyczne przetwarzanie obrazów kolorowych bez ingerencji użytkownika. W przypadku pozostałych urządzeń niezbędne jest dokupienie oddzielnego RGB Splitter'a lub stosowanie filtrów selektywnych. W przypadku przetwarzania obrazu z kamery rozwiązanie ostatnie umożliwia wprawdzie uzyskanie kolorowego obrazu w postaci cyfrowej, jednakże czas niezbędny dla uzyskania pożądanego efektu znacznie się wydłuża. W przypadku natomiast gdy posiadamy gotowy sygnał np. z magnetowidu i chcemy uzyskać kolor po przetworzeniu, dokupienie RGB Splitter'a staje się koniecznością.
W zestawie Digi Tiger II znajduje się oprócz samego urządzenia, którego obudowa jest zgodnie z nazwą ozdobiona „tygrysimi" prążkami, także odpowiedni zasilacz (urządzenie nie jest zasilane z komputera) sieciowy, specjalny kabel służący do połączenia digitizera z komputeren, instrukcja oraz dyskietka z oprogramowaniem. Odpowiedni program instalacyjny przewidziany jest dla użytkowników dysków twardych. Pozostali użytkownicy powinni wykonać sobie kopię oryginalnej dyskietki i z niej korzystać. Na dyskietce znajdują się dwie wersje programu obsługującego digitizer. Jedna przeznaczona jest dla „normalnych" Amig a druga to tzw. wersja Turbo dla Amigi 3000 oraz wszystkich pozostałych wyposażonych w turbo-karty. Możemy również wybrać w jakim języku (angielskim czy niemieckim) program będzie się z nami komunikował. Oprócz samego programu do obsługi digitizera na dyskietce znajduje się również kilka gotowych obrazków demonstracyjnych oraz program umożliwiający stworzenie tzw. Dia-Show czyli wyświetlenia obrazków w określonej kolejności. Na płycie czołowej urządzenia znajdują się cztery pokrętła regulacyjne umożliwiające regulację synchronizacji, jaskrawości, kontrastu oraz nasycenia kolorów. Podłączamy wszystko co potrzeba, to znaczy zasilacz, kamerę i komputer do digitizera po czym uruchamiamy program i zabieramy się do testów. Zgodnie z tym wybieramy tryb TEST i od razu widzimy na ekranie..., że nic nie widzimy! Aha! Zapomnieliśmy zdjąć przykrywkę przesłaniającą obiektyw kamery. Poprawiamy ten kardynalny błąd i tym razem jest! Obraz na ekranie komputera nie





Disk Resolution Digitize

320	x	256	1
320	x	512	2
640	x	256	3
640	x	512	4
352	x	280	5
352	x	560	6
704	x	280	7
704	x	560	8

Digitalizować możemy w każdej rozdzielczości!



Również takie efekty nie są problemem !

spełnia co prawda jeszcze naszych oczekiwań, ale łapiemy szybko za pokrętła regulacyjne i... tu spora niespodzianka. Obraz w trybie TEST jest przetwarzany w czasie około jednej (!) sekundy. Pozwala to na bezproblemowe ustawienie pokręteł regulacyjnych w żądane położenie. Mam tu na myśli fakt, że efekt każdej zmiany położenia pokręteł jest niemalże natychmiast widoczny na ekranie. Przy okazji „kręcenia gałkami" wyszły na jaw ciekawe możliwości uzyskania dodatkowych efektów poprzez różne ustawienia kontrastu i jaskrawości. Przyzwyczajeni dotychczas do urządzeń typu Digi-View, gdzie na efekty digitalizacji trzeba było czekać kilkanaście do kilkadziesiąt sekund, musimy przyznać, że jest to spory skok naprzód. Dodatkowe ciekawe efekty można uzyskać wybierając tryb PSEUDO-COLORS. W trybie tym poszczególnym odcieniom szarości przypisane są przypadkowo dobrane kolory z całej palety barw Amigi. Inną ciekawostką jest tryb ANTIQUE, po wybraniu którego możemy uzyskać efekt pożółkłego ze starości zdjęcia z albumu pradziadka. Zmieniamy tryby rozdzielczości i okazuje się, że w każdym z nich Digi Tiger bije na głowę konkurencję pod względem szybkości. Przy wyższych rozdzielczościach różnice uwidaczniają się jeszcze bardziej. Około 5 sekund trwa przetworzenie obrazu o wymiarach 640x512 punktów. Zaczynamy testować kolor. Program pozwala nam na skorzystanie w tym celu zarówno z zainstalowanego wewnątrz obudowy RGB Splitter'a, jak również (np. w przypadku gdy dysponujemy tylko kamerą czarno-białą) ze specjalnych filtrów umieszczanych kolejno przed obiektywem kamery. W pierwszym przypadku sprawa jest znacznie prostsza. Urządzenie automatycznie przetwarza kolejno obraz trzy razy w kolorach podstawowych, a następnie program tworzy nam według zadanych parametrów obraz kolorowy korzystając z zapamiętanych wcześniej obrazów RGB. Mamy możliwość wybrać spomiędzy wszystkich trybów dostępnych na Amidze. W przypadku, gdy wybierzemy tryb HAM, można jeszcze dodatkowo ustawić żądaną ostrość konturów. W tym miejscu oprogramowanie dostarczane do digitizerów typu Digi-View czy Deluxe View wydaje się być odrobinę bardziej dopracowane i dające większe możliwości obróbki po digitalizacji (np. Dynamic Hires w Digi-View). Niemniej jednak efekty uzyskane przy pomocy oryginalnego oprogramowania Digi Tiger'a są w pełni zadowalające. Obrazy są ostre i wyraźne, a barwy odtwarzane z wysoką wiernością. Gotowy obraz można nagrać na dysk w formacie IFF, a następnie dokonać dodatkowej obróbki przy użyciu dowolnego programu graficznego. Porównując z innymi digitizerami, należy stwierdzić, że DT II daje nam (przy podobnej cenie) podobne efekty końcowe co Digi- czy Deluxe View, natomiast komfort pracy (przede wszystkim szybkość) jest na zdecydowanie (kilka, do kilkanaście (!) razy) wyższym poziomie. Podsumowując nasz test musimy stwierdzić, że w pełni zgadzamy się ze sloganem reklamowym producenta, który w wolnym tłumaczeniu brzmi: Jeżeli digitizery przetwarzające obraz w czasie rzeczywistym są dla ciebie zbyt drogie a wszystkie pozostałe zbyt powolne, pozostaje ci tylko...

Digi Tiger II.

SD!

Urządzenie: video digitizer „Digi Tiger II"
Producent: Klaus D. Tute Soft-, Art- und Hardware
Dostawca: Silver Dream !'s
Cena: około 580,- DM

+ duża szybkość przetwarzania obrazu
+ wysoka jakość uzyskiwanych efektów
+ sprawdzona zgodność programowa z Kickstartem 1.2, 1.3, 2.0
+ wszystkie rozdzielczości dostępne już przy 1MB RAM
- ubogie możliwości ingerencji w obróbkę po digitalizacji

Ocena KEBAB'a: (w skali od 1 do 6) - 5 (bardzo dobra).

Słowniczek:

Composite Video - złożony sygnał wizyjny. Analogowy sygnał, w którym zakodowane są jednocześnie wszystkie elementy niezbędne do odtworzenia kolorowego obrazu np. na ekranie monitora tj. luminancja niosąca informację o jaskrawości poszczególnych punktów, chrominancja odpowiadająca za kolor oraz synchronizacja. Standard ten stosowany jest aktualnie w większości amatorskiego sprzętu Video ze względu na możliwość korzystania z jednego tylko toru elektrycznego. Do celów profesjonalnych stosowane są inne standardy ze względu na duże straty jakości podczas procesu kodowania/dekodowania sygnału Composite Video.

RGB Video - Sposób przekazywania obrazów przy użyciu czterech (lub więcej) niezależnych sygnałów. Trzy z nich to poszczególne sygnały kolorów składowych, (R)ed - czerwony, (G)reen - zielony i (B)lue - niebieski ,a następne to sygnały odpowiadające za synchronizację. System ten wykorzystuje się m.in. do podłączenia standardowego monitora do Amigi. Zaletą systemu RGB jest wysoka jakość przekazywanych obrazów (brak procesu kodowania/dekodowania), wadą natomiast konieczność przekazywania co najmniej czterech niezależnych sygnałów.

Component Video - coś pośredniego pomiędzy Composite a RGB. W tym systemie przekazywane są dwa niezależne sygnały tzw. komponentowe. System ten posiada wiele odmian i jest szeroko rozpowszechniony w profesjonalnych (np. Betacam SP) i półprofesjonalnych (S-VHS, Hi-8) urządzeniach do zapisu i obróbki obrazu.

RGB Splitter - urządzenie umożliwiające rozbicie (split) danego sygnału Composite Video na składowe sygnały RGB. Specjalistyczne układy tego typu znajdują się w każdym kolorowym telewizorze lub monitorze wyposażonym w wejście Composite Video.

Masterseka-HELP - c.d. ze strony piątej

wi [nazwa pliku] : Zapis pliku.
wo [nazwa pliku] : Zapis obiektu.
ws [numer napędu] : Zapis sektorów.
wt [numer napędu] : Zapis track'ów.
y Załadowanie plików dla „Extern" (patrz dalej).

Opcje assemblacji:
E Wydruk na drukarkę.
H Zatrzymanie po zapełnieniu strony przy listowaniu.
O : Optymizacja
P Wydruk na drukarkę.
V Wydruk na ekran.

Kombinacje z klawiszem Amiga w edytorze:
a : Szukaj poprzedniego ciągu znaków.
A Asemblacja (bez dodatkowych opcji)
b : Markuj blok.
c : Kopiuj blok do bufora.
i : Wstaw blok z bufora.
k Wytnij linię
q Wyjście z edytora.
Q . Asemblacja i wykonanie (skok)
r Wymień następny ciąg znaków
R : Wymień ciąg znaków.
s : Szukaj następnego ciągu znaków
S : Szukaj ciągu znaków.
x : Wytnij blok.
y : Wytnij do końca linii.
, . Wstaw średniki na początku linii umieszczonych w bloku
: Usuń średniki

Kombinacje z klawiszem Amiga w debuggerze:
o : Kasowanie starego i ładowanie nowego source'a
q : Wyjście z MasterSeki.

Kombinacje z klawiszem CTRL:
A : Kursor jeden ekran w górę
B : Markuj blok
C : Wytnij blok.
D : Skasuj do końca linii
E Kursor do końca linii.
F : Wstaw blok.
K : Wytnij linię.
L . Wymień w bloku litery na małe.
O : Wstaw linię
P Wstaw blok
Q : Kursor na początek linii
R : Odwróć blok
S . Kursor 100 linii w górę
T : Kursor na początek, lub koniec source'a.
U : Wymień w bloku litery na duże.
W : Zapisz blok na urządzeniu zewnętrznym
X : Kursor 100 linii w dół.
Y : Kursor jeden ekran w dół
Z : Kursor jeden ekran w dół.



Kombinacje z klawiszem ALT:
<kursor w prawo> : Skok do wskazanej etykiety.
<kursor w lewo> : Skok powrotny do odwołania etykiety

Wybrane dyrektywy:
align <liczba> : Przesunięcie asemblacji do adresu podzielnego przez <liczba>.
blk.b/w/l <wielkość>,<dana> : Tworzy blok danych wielkości <wielkość> i wypełnia go wartością <dana>. Wielkość bloku mierzona jest odpowiednio w bajtach/słowach/długich słowach.
dc.b/w/l <dana1,[dana2,...]> : Wstawia dane o rozmiarze bajtu/słowa/długiego słowa.
ds.b/w/l <wielkość>.<dana> : Patrz „blk"
endm : Kończy makrodefinicję.
equ : Nadaje symbolowi wartość.
even : Przesunięcie asemblacji do adresu parzystego.
end : Kończy asemblację. Domyślnie na końcu każdego source'a
>extern <nazwa pliku>,<adres> : Po zasemblowaniu i wykonaniu polecenia „y" plik <nazwa pliku> zostanie załadowany do pamięci pod adres <adres>.
incbin <nazwa pliku>.<adres> : Dołącza podczas asemblacji plik <nazwa pliku> pod adres <adres>
load <adres> : Asembuje source'a pod adres <adres>.
macro : Rozpoczyna makrodefinicję (patrz przykład).
odd : Przesunięcie asemblacji do adresu nieparzystego
org <adres> . Asembluje organizując source'a od adresu <adres>.
section <nazwa>,<typ> : Organizuje sekcję określonego typu. Dozwolone są:
- code : kod, czyli program w języku maszynowym.
- code_c : kod, wymuszenie do pamięci CHIP.
- bss : blok zer.
- bss_c : blok zer, wymuszenie do pamięci CHIP.



Przykład makrodefinicji:

```
Blit   macro
btst   #14,$dff002
dc.w   $66f6
endm
```

Ta krótka procedurka czeka na skończenie pracy przez Blitter'a

Krzysztof Kobus

64

Assembler na C-64

(odc.5)

Podczas poprzedniego naszego spotkania rozpoczęliśmy omawianie najważniejszych procedur systemu operacyjnego KERNAL. Tym razem poznamy jeszcze kilka z nich aby w następnym odcinku zabrać się już do bardziej efektownych rzeczy. Na początek coś zupełnie odwrotnego do omawianej ostatnio procedury CHROUT. Dwie procedury o nazwach GETIN oraz CHRIN (patrz tabela skoków z poprzedniego odcinka) znajdujące się odpowiednio pod adresami $FFE4 i $FFCF w tabeli. Służą one do wprowadzania danych z urządzenia zewnętrznego. Standardowo, tj. po wykonaniu RESET'u zewnętrznym urządzeniem wejściowym jest klawiatura. Tak, tak! Klawiatura z punktu widzenia systemu operacyjnego wcale nie jest częścią składową komputera, lecz urządzeniem zewnętrznym do niego podłączonym. W praktyce również nie mija się to z prawdą. Jeżeli ktoś próbował kiedyś rozmontować swojego (lub np. kolegi) "komodorka" to wie, że klawiaturę podłącza się do płyty głównej za pomocą specjalnego złącza, a to, że mieści się ona w tej samej obudowie co komputer to tylko jedna z możliwości. Wracając do rzeczy! Jak wspominałem w/w procedury służą do WPROWADZANIA danych. Chwilowo nie będziemy rozdrabniać się na inne urządzenia zewnętrzne (zbiory dyskowe, modemy itd.) i zajmiemy się wyłącznie klawiaturą. Z pewnością zdarzyło wam się nie raz, że pisany przez was program, np. w BASIC'u, musiał być zatrzymany do momentu wciśnięcia jakiegoś klawisza, ot choćby po to, aby jego użytkownik mógł przeczytać wydrukowany na ekranie komunikat. Zazwyczaj stosuje się właśnie taką metodę pisząc na końcu "Press any key to continue..." lub "Wcisnij dowolny klawisz...". Celowo napisałem "wcisnij" a nie "wciśnij" gdyż do tego jak zrobić polskie znaki jeszcze nie doszliśmy (ale szukajcie, a znajdziecie... w KEBAB'ie oczywiście!). Użytkownik naszego programu po przeczytaniu tekstu naciska jakiś klawisz (najczęściej SPACE) i program zaczyna działać dalej. Tak naprawdę to działał on cały czas choć nic nie działo się na ekranie. Jak taki efekt uzyskać? W BASIC'u najczęściej stosowane są dwa chwyty:

```
100 POKE 198,0:WAIT
198,1
```

lub

```
100 GET ZZ$:IF ZZ$=""
THEN GOTO 100
```

Pierwsza metoda wykorzystuje tzw. wskaźnik bufora klawiatury i jest w naszym przypadku mało przydatna jako przykład. Druga natomiast, to niemal (zaznaczam: niemal!) bezpośrednie wywołanie procedury GETIN z tabeli skoków. Jak działa ta procedura? W przypadku klawiatury jako aktualnego urządzenia I (wejścia), pobiera ona do akumulatora jeden znak z wspmnianego już bufora klawiatury, a to jest właśnie to o co nam chodzi. Musimy pamiętać tylko o jednym! Odczyt klawiatury i umieszczenie odpowiednich znaków w w/w buforze to zadanie procedur przerwań systemu operacyjnego. Czyli, aby korzystać z jego usług nie możemy zablokować przerwań systemowych np. komendą SEI. Ale, ale... o czym ja mówię, to wszystko to jeszcze nie nasz etap. Na razie nie używamy w ogóle takich komend! Zatem wzorując się niemal dokładnie na drugim przykładzie BASIC'owym napiszmy to samo w języku asemblera.

```
A5000  JSR  $FFE4
A5003  BEQ  $5000
A5005  RTS
```

Co my tu mamy? JSR $FFE4, a więc skok do podprogramu - procedury GETIN, poprzez tabelę oczywiście. Następnie BEQ $5000 tzn. jeżeli wynik ostatniej operacji (a mamy go w akumulatorze) jest równy zero, to skaczemy ponownie do $5000 i po raz kolejny wywołujemy procedurę GETIN. Prawda, że podobne do BASIC'a? Tam też było "IF ZZ$="" THEN...", czyli jeżeli po wywołaniu funkcji GET ZZ$ zmienna ZZ$ zawierała pusty łańcuch (dwa znaki cudzysłowia oznaczają w BASIC'u tzw. łańcuch pusty - ang.: empty string), to wywoływaliśmy ją ponownie (... GOTO 100). W przypadku natomiast, gdy uzyskaliśmy w akumulatorze wynik różny od zera, to program przejdzie do następnej linii... a RTS w naszych dotychczasowych przykładach to wszyscy wiemy co oznacza...! (Mam nadzieję !?) Spróbujmy to uruchomić: piszemy szybko SYS 5*4096 i... nic się nie dzieje! Kursor zniknął, a komputer "wisi". Ze złością uderzamy w klawisze... NO! Pomogło! Znowu pojawił się kursor... zaraz, zaraz... przecież to właśnie tak miało być! Aby upewnić się, że komputer wcale nie "zawisł" proponuję troszkę inny test:

```
A5000  LDA  #$00
A5002  STA  $D020
A5005  JSR  $FFE4
A5008  BEQ  $5005
A500A  LDA  #$0E
A500C  STA  $D020
A500F  RTS
```

Tym razem zmieniamy kolor ramki przed i po wciśnięciu klawisza ale

zamiast tego możemy zrobić coś innego np. wyświetlić fragment tekstu przed wciśnięciem klawisza (od poprzedniego odcinka wiemy jak to robić), a drugi fragment tekstu po wciśnięciu. Dotychczas sprawdzaliśmy czy nie został wciśnięty jakikolwiek klawisz, ale procedura GETIN potrafi nam również podać jaki klawisz został wciśnięty. Po powrocie z procedury mamy w akumulatorze kod PETASCII znaku odpowiadającego klawiszowi lub kombinacji klawiszy wciśniętych w momencie wywołania procedury. Spróbujmy na przykład zatrzymać program do momentu naciśnięcia klawisza SPACE (nie wtajemniczonym przypominam, że jest to ten najdłuższy klawisz na klawiaturze). Napiszmy zatem:

```
A5000  JSR  $FFE4
A5003  CMP  #$20
A5005  BNE  $5000
A5007  LDA  #$93
A5009  JMP  $FFD2
```

Tym razem mamy tu kilka nowości. JSR $FFE4 to powinno być jasne ale "CMP #$20"? Tak, to coś nowego! Jak zwykle podam pełne, angielskie brzmienie wyrażenia od którego pochodzi ten skrót. Chodzi tu o zwrot (C)o(MP)are Accumulator. Po polsku znaczy ono po prostu "porównaj akumulator". Dobrze! Ale z czym porównujemy ten akumulator? Dokładnie porównujemy zawartość akumulatora z wartością określoną przez operand (patrz również poprzednie odcinki kursu). Znaczek "#" określa nam, co wiemy już z naszego kursu, tzw. adresowanie bezpośrednie (lub "natychmiastowe") od angielskiego "Immediate addresing". To znaczy, że porównujemy zawartość akumulatora z wartością występującą bezpośrednio po znaczku "#". W naszym przykładzie mamy po znaczku "#" wartość $20 i to jest to, z czym będziemy porównywać akumulator. Spoglądamy do tabeli ASCII (np. w instrukcji obsługi komputera) i odszukujemy tam liczbę $20, a właściwie jej dziesiętny odpowiednik (32) i co widzimy? Widzimy, że jest to kod odpowiadający znakowi spacji. Aha! To znaczy, że wywołujemy procedurę GETIN, a po powrocie z niej mamy w akumulatorze wartość odpowiadającą kodowi PETASCII wciśniętego aktualnie klawisza. Ewentualnie, gdy żaden klawisz nie jest wciśnięty, otrzymujemy w akumulatorze wartość $00. Teraz porównujemy to, co uzyskaliśmy w wyniku wywołania GETIN z wartością odpowiadającą za klawisz spacji ($20). I Tu przychodzi czas na wyjaśnienie dlaczego w poprzednich odcinkach tłumacząc znaczenie komend BEQ, czy BNE pisaliśmy słowo "zero" w nawiasie. Otóż komend tych używamy nie tylko wtedy, gdy interesuje nas czy wynik ostatniej operacji jest równy bądź różny od zera. Również wtedy, gdy porównujemy ze sobą dwie liczby i zależnie od tego czy są one równe czy różne, podejmujemy odpowiednie działanie. Wynika to z tego, że procesor wykonując komendę CMP (a również CPX czy CPY - odpowiedniki CMP odnoszące się do rejestrów X i Y), wykonuje wewnętrznie operację odejmowania liczby porównywanej od zawartości akumulatora (bądź rejestru X czy Y). Natomiast wynik tego odejmowania nie jest nigdzie przechowywany, jedynie procesor wie czy wynik tego odejmowania był równy zero (gdy liczby są sobie równe) czy też nie (jeżeli liczby były różne). Dlatego właśnie używaliśmy nawiasów, bo podstawowym zastosowaniem tych komend jest podejmowanie decyzji po porównaniu. Jednakże specyfika ich działania pozwala na znacznie więcej zastosowań. Ale odeszliśmy chyba zbyt daleko od naszego przykładu. Wracamy zatem. Po porównaniu akumulatora z wartością $20 podejmujemy decyzję. BNE - to (B)ranch if (N)ot (E)qual, czyli skocz jeśli nie są równe (pod adres $5000). Tam ponownie wywołujemy GETIN. I tak w kółko, aż do momentu, gdy przy kolejnym skoku pod $FFE4 będzie wciśnięty klawisz spacji. Wtedy po powrocie z procedury w akumulatorze znajdzie się oczekiwana przez nas wartość $20 i wtedy też, zamiast skakać znowu pod $5000, procesor wykona kolejną komendę. Zestaw LDA #$93 i JSR $FFD2 to już wszyscy znamy. Tu jednak mamy nie "JSR", a "JMP" i na końcu nie mamy znanego nam już dobrze RTS'a. Dlaczego? Otóż wszystkie udokumentowane procedury systemu operacyjnego są to podprogramy, które prędzej lub później kończą się rozkazem powrotu z podprogramu... tak, właśnie RTS'em. Zatem każdą kombinację typu:

```
JSR  $XXXX
RTS
```

możemy zastąpić jednym rozkazem

```
JMP  $XXXX
```

Oszczędzamy w ten sposób jeden (!) bajt, co może się kiedyś przydać (naprawdę!), ale tracimy sporo na przejrzystości programu. Dlatego też, w naszych przykładach będziemy stosować jednak pierwszy sposób. Uff! Przebrnęliśmy przez ten krótki przykładzik. Domyślni Czytelnicy pewnie już domyślili się, że w ten sam sposób możemy sprawdzić dowolny klawisz np. (T)ak lub (N)ie, albo (Y/N) itd. Ale w BASIC'u mamy więcej możliwości wprowadzania danych. Funkcja GET służy zazwyczaj do wprowadzania pojedynczych znaków. Owszem, można ją przy użyciu odpowiednich pętli użyć do wprowadzania całych ciągów liter, lecz nie stosujemy tego typu trików, bo mamy gotową komendę INPUT. A skoro funkcję GET możemy w języku asemblera zasymulować korzystając z jednej tylko procedury systemu operacyjnego, to może i INPUT da się zrealizować w podobny sposób? Owszem! Służy do tego druga z wymienionych dzisiaj procedur KERNAL'a tzn. CHRIN. Adres tej procedury w tabeli skoków to $FFCF. Obie procedury działają identycznie w przypadku wszystkich urządzeń wejścia za wyjątkiem klawiatury. To znaczy, że jeżeli będziemy odczytywać bajty (znaki) np. ze stacji dysków to nie ma znaczenia, której procedury będziemy używać. Natomiast w przy-

padku odczytywania znaków z klawiatury obie procedury działają inaczej. Pierwszą już poznaliśmy. Działanie drugiej proponuję poznać na przykładzie, który znajduje się w dziale Listingi. Składa się on z dwóch części. Obie należy wpisać korzystając z odpowiedniego debuggera. Jest to pierwszy dość "poważny" przykład obrazujący to, że w języku asemblera można równie szybko jak w językach wyższego rzędu tworzyć programy i procedury o dość wysokim stopniu złożoności. Warunkiem jest dobra znajomość systemu operacyjnego. Po wpisaniu obu części programu uruchamiamy go jak zwykle rozkazem SYS 5*4096 z BASIC'a lub odpowiednią komendą G (lub J) debuggera. Przyjrzyjmy się temu programowi bliżej. Pierwsze pięć linijek znamy już wszyscy z poprzednich odcinków. Przejdźmy zatem do następnych. Kolejne sześć linijek to znana już również z odcinka czwartego procedurka drukująca na ekranie ciąg znaków, których kody znajdują się począwszy od adresu $5080. Począwszy od adresu $501A rozpoczyna się nowa procedurka odwołująca się do CHRIN ($FFCF) w tabeli skoków. Domyślamy się, że to właśnie jest realizacja funkcji INPUT z poziomu języka maszynowego. I zgadza się! Ale jak to działa? Popatrzmy! Na początku mamy LDX #$00, czyli wartość $00 do rejestru X. Dalej JSR $FFCF, to znaczy od razu skok do ROM'u. Procedura CHRIN jest tak skonstruowana, że pierwsze jej wywołanie powoduje (jeżeli aktualnym urządzeniem wejściowym jest klawiatura) pojawienie się migającego kursora, a następnie wciśnięcie każdego klawisza powoduje to, że odpowiadający mu kod zostaje umieszczony w tzw. buforze klawiatury tj. obszarze pamięci zarezerwowanym przez system do czasowego przechowywania kodów wciskanych klawiszy. Procedura również, oprócz umieszczenia tego kodu we wspomnianym buforze wysyła go na ekran. Dzięki temu możemy widzieć co aktualnie piszemy. Wszystko to dzieje się tak długo, dopóki nie wciśniemy klawisza RETURN. Wtedy to, procedura wraca do miejsca wywołania, przekazując w akumulatorze pierwszy (nie ostatni) z kodów umieszczonych w buforze klawiatury. Każde następne wywołanie tej procedury zwraca nam w akumulatorze kolejny kod. W momencie, gdy pobierzemy już kod ostatniego wciśniętego klawisza, a będzie to zawsze $0D odpowiadający klawiszowi RETURN, kolejne wywołanie procedury rozpocznie cały proces od nowa. Zatem, aby uniknąć więcej niż jednego pobierania łańcucha tekstowego, zaraz po powrocie z procedury sprawdzamy komendą CMP, czy przypadkiem nie jest to już ostatni kod ($0D). Jeżeli tak, to od razu skaczemy pod adres $5029 i wykonujemy dalszy ciąg programu. Jeżeli nie to musimy gdzieś sobie zapamiętać wszystkie znaki, które zostaną nam przekazane podczas kolejnych wywołań CHRIN. W tym celu postanowiliśmy umieszczać je od adresu $5100 w górę. Robi to nam komenda STA $5100,X. INX to nie muszę już chyba tłumaczyć. A BNE wykonuje skok (jeżeli tylko rejestr X nie "przeskoczył") pod adres $501C, gdzie po raz kolejny wykonujemy JSR $FFCF. W momencie gdy uzyskamy już wartość $0D co oznacza koniec łańcucha, rozpoczynamy drukowanie na ekranie wszystkiego co potrzeba. Lecz jeszcze zanim to uczynimy musimy wpisać na koniec naszego dopiero co zapamiętanego (pod $5100) łańcucha znacznik jego końca. Wcześniej był to bajt $0D. My jednak przyjmiemy sobie, że znacznikiem końca łańcucha będzie zawsze bajt $00. Zatem przesyłamy zawartość akumulatora do rejestru Y (TAY), i ładujemy na to miejsce $00. Następnie umieszczamy ten bajt na końcu naszego łańcucha. Komenda TYA (Transfer Y to Accumulator) przesyła nam na powrót przechowaną czasowo w rejestrze Y wartość $0D z powrotem do akumulatora, a kolejne trzykrotne wywołanie procedury CHROUT spowoduje, że kursor przeskoczy o trzy linie w dół. Teraz już drukujemy kolejno na ekranie teksty wklepane uprzednio pod odpowiednie adresy oraz nasz, podany z klawiatury łańcuch i na końcu grzecznie powracamy do punktu wyjścia komendą RTS (oczywiście po wciśnięciu dowolnego klawisza $506D - $5070). Ktoś mógłby powiedzieć: przecież to wszystko mogę zaprogramować w BASIC'u! Owszem! Ale zanim przejdziemy do takich rzeczy, których w BASIC-'u nie da się zrobić, należy poznać podstawy, bez których w pewnym momencie możemy ugrzęznąć podczas programowania większych programów. Poza tym, nie należy do najlepszych metod programowanie części programu w BASIC'u, a części tylko w MC. Po prostu nie zawsze się ten sposób sprawdza. Ja ze swej strony zapraszam już do lektury następnego odcinka gdzie poruszymy jak zwykle sporo nowych tematów. A na zakończenie jeszcze jeden króciutki przykład. Jest to najczęściej stosowana przez koderów metoda na sprawdzenie czy nie został wciśnięty klawisz spacji. W poprzednich przykładach robiliśmy to korzystając z procedur systemu operacyjnego. Tym razem wykorzystamy cechy sprzętowe naszego "komcia".

```
A5000 LDA $DC01
A5003 CMP #$EF
A5005 BNE $5000
A5007 RTS
```

W następnym odcinku wyjaśnimy co oznacza to tajemnicze $DC01. A póki co, życzę ciekawych eksperymentów z dzisiejszymi przykładami.

SD!

Do zapamiętania:
- procedury CHRIN oraz GETIN
- sposób przechowywania podawanych przez procedurę CHRIN znaków
- komendy CMP, TAY, TYA, BEQ, BNE
- odczyt pojedynczych klawiszy



Grafika wektorowa

odrobina teorii

Od dłuższego już czasu wszystkich, bardziej lub mniej znanych koderów, zarówno w kraju jak i za granicą ogarnął szał grafiki wektorowej. W setkach powstających programów demonstracyjnych mamy okazję podziwiać najróżniejsze bryły, statki kosmiczne, oraz inne obiekty animowane przy pomocy tej właśnie techniki przetwarzania obrazu. Często bywa, że zrobienie wektorówki jest pewnego rodzaju sprawdzianem dla kodera, podobnie jak pierwsze przepłynięcie marynarza przez równik. Osobom posiadającym już pewne doświadczenie w kodowaniu, chciałbym na łamach Kebaba przybliżyć sposób realizacji procedur grafiki wektorowej.

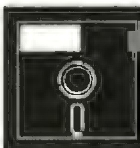
Dlaczego grafika wektorowa?

Opisanie przestrzeni nas otaczającej, tudzież poszczególnych jej obiektów za pomocą współrzędnych wierzchołków, czyli wektorów, daje użytkownikowi ogromne możliwości przetwarzania obrazu, a przede wszystkim ogromną dokładność i precyzję otrzymanych wyników. Dlatego właśnie grafika wektorowa znajduje zastosowanie we wszystkich programach wspomagających projektowanie. Pozwala na dowolne skalowanie obrazu, jego translację, obroty, czyli możliwość oglądania obiektu z dowolnej strony, także od wewnątrz. Na przykład współczesne programy do sporządzania projektów domów, działające w oparciu o grafikę wektorową wraz z tzw. ray-tracingiem, tj. śledzeniem promieni światła oraz analizą jego odbicia, załamania i rozproszenia, umożliwiają przyszłemu mieszkańcowi na dokładne zaznajomienie się z wyglądem i rozmieszczeniem pokoi oraz innych pomieszczeń. Na tym etapie można stworzyć taki projekt, który zarówno będzie odpowiadał właścicielowi, jak i będzie możliwy do wykonania pod względem technicznym. Przyszły mieszkaniec ma możliwość „marudzenia" przesuwając płynnie ściany, zwiększając jedne pomieszczenia kosztem drugich, oczywiście w granicach rozsądku. Wreszcie ostateczny wygląd zostaje zaakceptowany i ostatecznie wykonany. Grafika wektorowa to oczywiście nie tylko projektowanie. Najlepszym tego przykładem może być Kebab, który właśnie czytacie. Zanim stał się takim, jak obecnie wygląda, całkowicie podczas składu zapisany był właśnie wektorowo. Wszystkie współrzędne każdej litery wielokrotnie były przeliczane i wreszcie za kolejnym razem osiągnęły taką postać jaką macie przed sobą. Również znak „Kebab" znajdujący się u dołu strony narysowany został na programie zapisującym rysunki za pomocą wektorów, dzięki temu niemal błyskawicznie można go wyginać, skalować, ustawiać względem danej perspektywy, naciągać na kulę i transformować na wiele różnych sposobów. Podobne przykłady można przytaczać bez końca. Współczesne projektowanie płytek drukowanych do obwodów elektronicznych, liczne programy symulacyjne - realizowane są właśnie przy pomocy techniki wektorowej, a o tym, że napisanie procedur przeliczających wierzchołki i transformujących obrazy nie jest takie trudne, przekonacie się za chwilę.

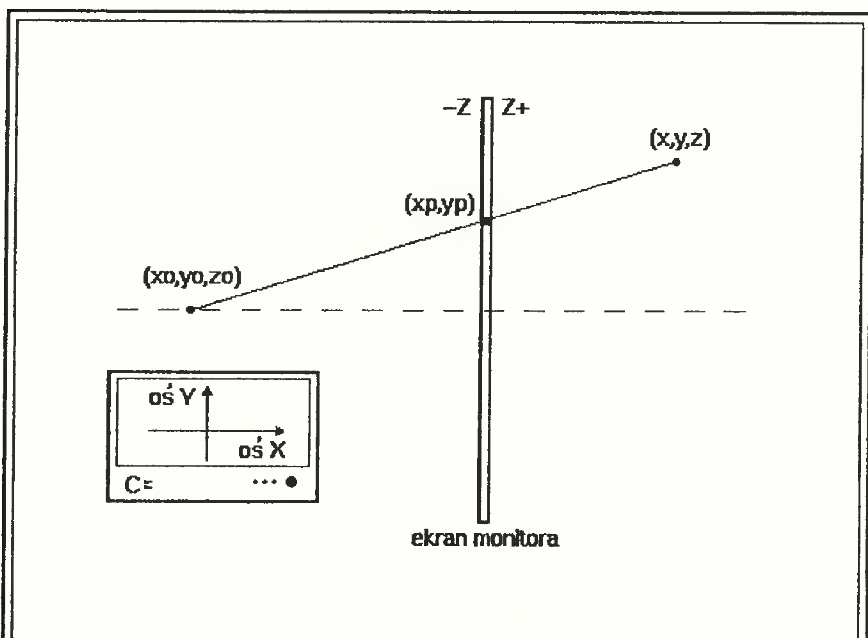


Trochę matematyki.

Realizacja procedur grafiki wektorowej zależy przede wszystkim od tego do jakich celów chcemy ją wykorzystać. Zdarza się czasem, że obliczenia sprowadzają się do transformacji przestrzeni dwuwymiarowej, czyli płaszczyzny (DTP). W takim przypadku stosujemy wzory takie, jakie poznaliście w szkole średniej, na zajęciach z geometrii (o ile w tym samym czasie nie byliście na przykład w kinie). Obecnie chciałbym przedstawić wektorówkę taką, jaką znacie z licznych programów demonstracyjnych i, aby nie zaśmiecać wam zbytnio w głowię tym razem ograniczę swoje wywody do zagadnień wektorówki „niewypełnionej". Zanim zaczniecie pisać jakiekolwiek procedury, zastanówcie się nad wprowadzeniem wyimaginowanej przestrzeni, w której będzie znajdował się obserwator oraz będą poruszały się obiekty. Dla uproszczenia i zarazem przyspieszenia obliczeń, najwygodniej jest zrealizować to w ten sposób, że monitor widziany od przodu zawierać będzie płaszczyznę XY, a prosta prostopadła do niego będzie osią Z naszego układu OXYZ. Przyjmijmy także, że punkty znajdujące się między nami a płaszczyzną monitora będą miały współrzędną Z ujemną, natomiast za płaszczyzną monitora dodatnią. Wynika z tego, że środek układu współrzędnych, czyli punkt (0,0,0) znajdował się będzie na płaszczyźnie monitora, a dokładnie na jej środku oraz wszystkie zdefiniowane obiekty będą miały współrzędną Z dodatnią.



Patrz również rysunek na następnej stronie.



Przyjmijmy taką organizację przestrzeni.

1. Translacja obiektu.
Translację, czyli przesunięcie o wektor, uzyskujemy przez dodanie do współrzędnych każdego punktu naszego obiektu, współrzędne wektora przesunięcia. Aby przesunąć więc punkt (x,y,z) o wektor [u1,u2,u3] należy wykonać:

x1=x+u1
y1=y+u2
z1=z+u3

Otrzymane (x1,y1,z1) są współrzędnymi punktu po przesunięciu.

2. Obrót obiektu.
Aby obrócić obiekt w przestrzeni w dowolny sposób należy złożyć obroty względem trzech osi: OX, OY, OZ. Zauważmy, że obracając obiekt względem osi OX jego współrzędna x nie zmienia się. Wystarczy więc przeliczyć jedynie współrzędne y oraz z obiektu. Podobnie obracając względem OY przeliczamy x i z, natomiast względem OZ x i y.
a) Obrót względem OX. Współrzedne y,z w obrocie wokół punktu (xo,yo,zo) o kąt A zmieniają się na współrzędne yr,zr według następujących wzorów:
yr=yo+(y-yo)*cos(A)+(z-zo)*sin(A)
zr=zo+(z-zo)*cos(A)-(y-yo)*sin(A)
Wynika z tego, że współrzędna xo punktu, wokół którego obracamy nie bierze udziału w tym obrocie.
b) Obrót względem OY. Analogicznie do powyższego obracając x,z wokół (xo,yo,zo) o kąt B otrzymujemy:
xr=xo+(x-xo)*cos(B)-(z-zo)*sin(B)
zr=zo+(z-zo)*cos(B)+(x-xo)*sin(B)
c) Obrót względem OZ. Wreszcie obracając x,y wokół (xo,yo,zo) o kąt C mamy:
xr=xo+(x-xo)*cos(C)+(y-yo)*sin(C)
yr=yo+(y-yo)*cos(C)-(x-xo)*sin(C)

3. Rzutowanie obiektu.
Rzutowanie polega na przekształceniu trójwymiarowego obrazu w ten sposób, aby mógł być przedstawiony na dwuwymiarowej płaszczyźnie monitora. W tym celu należy każdy punkt naszego obiektu o współrzędnych (x,y,z) przekształcić na punkt o współrzędnych (xp,yp) znajdujący się na ekranie monitora. Realizują to wzory:
xp=xo+((xo-x)*zo)/(z-zo)
yp=yo+((yo-y)*zo)/(z-zo)
przy czym współrzędne (xo,zo,zo) odpowiadają za położenie obserwatora, czyli ustawienie perspektywy. Należy pamiętać, aby rzutowanie wykonywać zawsze jako ostatnie, gdy wszystkie inne transformacje zostały już wykonane.
Przekształceń przestrzeni istnieje jeszcze bardzo wiele. Koderzy w programach demonstracyjnych ograniczają się przede wszystkim do opisanych powyżej. Sprytnie je składając otrzymują bardzo ciekawe efekty, niekiedy przypominające nawet krótkie filmy animowane. Proponuję teraz na podstawie powyższych wzorów napisać procedury grafiki wektorowej. Nie musicie od razu pisać ich w assemblerze jednak pamiętajcie, że właściwie tylko w tym języku uzyskacie największą efektywność. Miłego kodowania (programowania?) życzy:

Krzysztof Kobus

Listy, listy, listy...

...Zwracam się... ... z kilkoma problemami i mam nadzieję, że uzyskam pomoc... ... zajmuję się filmowaniem video-camerą. Chciałbym zastować Commodore 64 (ze standardowym wyposażeniem tzn. tylko klawiatura, zasilacz i magnetofon) do wykonywania czołówek, napisów itp. poprzez magnetowid na kasety... ... z instrukcji obsługi niewiele wynika (bardzo dokładnie przepisywałem programy z instrukcji do C-64 i nic z tego nie wychodziło !). Czy posiadacie takie opracowania, aby można było robić barwne, ruchome napisy przesuwające się z dołu do góry lub w formie "przewracania kartek", lub jeszcze inne "bajery", które można by było zastosować? Czy można coś takiego nabyć? Jaka jest tego cena?

Jerzy Własiuk

Szanowny Panie,
poruszył Pan w swoim liście co najmniej dwie istotne sprawy. Zacznę może od tej prostszej. Mam na myśli programy z instrukcji obsługi Są to krótkie przykłady w języku BASIC, których celem jest przedstawienie użytkownikowi podstawowych możliwości sprzętowych C-64. Na pewno nie są to żadne wyrafinowane programy umożliwiające np. wykonywanie płynnych efektów animacyjnych, o które Panu chodzi. Uzyskanie tego typu "bajerów" wymaga stosowania języka maszynowego. Inną rzeczą jest fakt, że prawdopodobnie (wnioskuję z Pańskiej wypowiedzi) nie udało się Panu tych przykładów uruchomić. Otóż

c.d. na str. 23



Mapa pamięci Amigi

(c.d.)

040 BLTCON0 W - A
042 BLTCON1 W - A
Rejestry kontroli Blittera. Poszczególne bity zerowego (BLTCON0) rejestru kontroli mają następujące znaczenie:
bity 15-12 (ASH3-ASH0) Bity przesunięcia dla źródła A. Jak wiadomo, na czterech bitach można zapisać liczby z zakresu od 0 do 15. O takie więc wartości możemy przesuwać obszar zdefiniowany jako źródło A.
bit 11 (USEA) Ustawiony oznacza użycie źródła A.
bit 10 (USEB) Ustawiony oznacza użycie źródła B.
bit 9 (USEC) Ustawiony oznacza użycie źródła C.
bit 8 (USED) Ustawiony oznacza użycie przeznaczenia (ang: Destination).
bity 7-0 (LF7-LF0) Bity określające jaką funkcję logiczną określoną na blokach pamięci Blitter ma realizować.
A oto rozpiska pierwszego (BLTCON1) rejestru kontroli:
bity 15-12 (BSH3-BSH0) Bity przesunięcia dla źródła B.
bity 11-5 Nie używane, lecz w celu zapewnienia kompatybilności z nowszymi modelami Agnus'a powinny zawsze być skasowane.
bit 4 (EFE) Włącza tryb „Exclusive" przy wypełnianiu.
bit 3 (IFE) Włącza tryb „Inclusive" przy wypełnianiu.
bit 2 (FCI) Używany przy wypełnaniu.
bit 1 (DESC) Ustawiony oznacza, że Blitter będzie pracował „od tyłu", tzn. kopiowanie bloków i operacje na nich odbywać się będą od adresów wyższych do niższych.
bit 0 (LINE) Ustawiony przełącza Blitter'a na tryb kreślenia linii. W tym trybie rejestry obsługujące opisywany układ posiadają odmienne znaczenie i zostaną dokładnie opisane w kolejnych odcinkach cyklu. Niniejszy opis rejestrów Blitter'a odnosi się do sytuacji, gdy bit ten jest skasowany.

044 BLTAFWM W - A
046 BLTALWM W - A
Maski dla pierwszego i ostatniego słowa bloku. Podczas operacji na pamięci Bliter wykona logiczne AND z wartością pierwszych słów pobranych z miejsca zdefiniowanego kanałem źródłowym A i wartością wpisaną do rejestru BLTAFWM.

048 BLTCPTH W - + A
04A BLTCPTL W - + A
Wskaźnik kanału źródła C. Do tych rejestrów wpisujemy adres bloku obszaru źródłowego C. Należy pamiętać, że adres ten musi wskazywać pamięć typu CHIP.

04C BLTBPTH W - + A
04E BLTBPTL W - + A
Analogicznie do BLTCPTH/L, dotyczy źródła B.

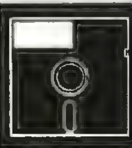
050 BLTAPTH W - + A
052 BLTAPTL W - + A
Analogicznie do BLTCPTH/L, dotyczy źródła A.

054 BLTDPTH W - + A
056 BLTDPTL W - + A
Wskaźnik obszaru przeznaczenia D. Wskazuje miejsce w pamięci CHIP, gdzie zostanie zapisany wynik operacji Blitter'a.

058 BLTSIZE W - A
Wielkość okna dla Blittera i start. Przy pomocy tego rejestru definiujemy wielkość obszaru, na którym będziemy operować Blitter'em.



bity 15-6 Definiują pionową wielkość okna. Łatwo wyliczyć, że przy pomocy tych dziesięciu bitów możemy ustalić największą pionową wielkość okna na wartość równą 2 do potęgi 10, czyli 1024.
bity 5-0 Definiują poziomą wielkość okna wyrażoną w słowach. Analogicznie do powyższego wyliczamy, że maksymalna wielkość pozioma okna wynosi 2 do potęgi 6 (mamy 6 bitów definiujących), czyli 64 słowa, co w przeliczeniu daje nam 64*16=1024 pixele.



Przy zapisie wielkości okna proponuję przyjąć następującą konwencję:
move.w
#[64*WYS]+SZER,$dff058
gdzie:
WYS - Wysokość okna wyrażona w liniach.
SZER - Szerokość okna wyrażona w słowach.
Iloczyn liczby 64 z wysokością okna zapewni nam przesunięcie wartości WYS na miejsce bitów 15-6.
Jak już wspomniałem na początku, rejestr ten odpowiada również za wystartowanie Bliter'a. Dlatego też, powinien być zapisywany, gdy wszystkie inne rejestry obsługujące ten układ zostały zainicjiowane.



060 BLTCMOD W - A
Modulacja Blitter'a dla kanału źródłowego C. Wartość wyrażoną w bajtach, wpisaną do tego rejestru, Bliter doda do wskaźnika kanału C, po skopiowaniu każdej linii bloku danych. Dzięki modulacji mamy możliwość kopiowania nie tylko obszarów pamięci o szerokości równej szerokości obrazka, lecz również małych okienek z niego wyciętych. Jeśli na przykład z ob-

razka o szerokości 320 (20 słów) pixeli chcemy skopiować okienko o szerokości 48 pixeli (3 słowa) modulację należy ustawić na wartość 34, ponieważ właśnie 34 bajty (to jest 34*8=272 pixele) należy dodać do aktualnego miejsca pamięci, aby trafić na miejsce początku okienka w następnej linii. W praktyce, najczęściej suma szerokości okienka i modulacji jest równa szerokości obrazka.

062 BLTBMOD W - A
Analogicznie do BLTCMOD, dotyczy kanału źródłowego B.

064 BLTAMOD W - A
Analogicznie do BLTCMOD, dotyczy kanału źródłowego A.

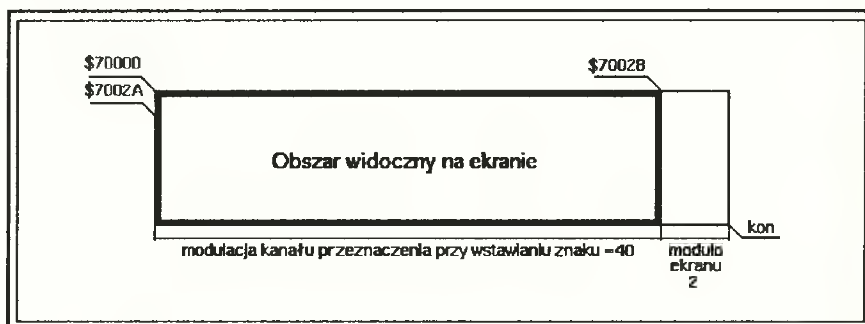
066 BLTDMOD W - A
Analogicznie do BLTCMOD, dotyczy kanału przeznaczenia D.

070 BLTCDAT W - % A
Rejestr danych dla źródła A. Praktycznie dostępny tylko dla wewnętrznych kanałów DMA, więc nie będzie nas zbytnio interesował.

072 BLTBDAT W - % A
Analogicznie do BLTCDAT, dotyczy źródła B.

074 BLTADAT W - % A
Analogicznie do BLTCDAT, dotyczy źródła A.

W ten właśnie sposób wygląda teoria Blitter'a. A jak jest w praktyce? No cóż, aby dobrze sobie to uzmysłowić zacznę od tego jak zorganizowana jest pamięć obrazu. Każdy zapewne domyśla się, że tak jak wszystkie pamięci składa się ona z komórek, z których każda zawiera bajt informacji, czyli osiem bitów. W przypadku wyświetlania obrazu dwukolorowego (tło - kolor zwany zerowym, i jeden kolor - zwany pierwszym), te osiem bitów pozwala na zapisanie danych dla wyświetlenia ośmiu punktów obrazu, zwanych pixelami. Jeśli procesor graficzny przeglądając pamięć stwierdzi, że dany bit jest ustawiony, odzwierciedla to pixelem na ekranie w kolorze pierwszym, jeżeli natomiast stwierdzi, że bit ten jest skasowany, odzwierciedla to na ekranie punktem w kolorze tła, czyli w kolorze zerowym. O tym, czy dany kolor będzie zielony, żółty czy jeszcze inny decydują specjalne rejestry, znajdujące się od adresu $dff180, którymi bliżej zajmiemy się przy innej okazji. Bazując na tych wiadomościach łatwo możemy wyliczyć, że dla wyświetlenia jednej tylko linii obrazu w trybie „Lo-Res", zajmującym 320 pixeli w poziomie, w dwóch kolorach, potrzebujemy 320 bitów pamięci, czyli 320/8=40 otrzymamy szukany adres: $70000+$28+$02=$7002A. A co z linią drugą? Proszę bardzo: adres linii poprzedniej (pierwszej) - $7002A, jej długość - $28, oraz modulo - $02. Dodajemy $7002A+$28+$02=$70054 i kolejny adres gotowy. Dokładnie przyglądając się temu mechanizmowi tworzenia obrazu, można dojść do wniosku, że część pamięci nie jest wyświetlana i znajduje się niejako za ekranem, a ściślej z jego prawej strony. Obrazuje to następujący rysunek:



bajtów. Wynika z tego, że adres początku każdej kolejnej linii w pamięci ekranu będzie sumą adresu linii poprzedniej i wartości 40. Dla przykładu załóżmy, że zerowa linia ekranu znajduje się pod adresem $70000, łatwo obliczyć, że dane w pamięci dla linii pierwszej znajdują się pod adresem $70000+$28=$70028 ($28=dec40), linii drugiej $70028+$28=$70050 ($50=dec80), itd... I rzeczywiście tak z reguły jest, ale być nie musi. Istnieje bowiem rejestr zwany „modulacją ekranu" (nie mylić z modulacją Blitter'a), do którego możemy wpisać wartość, która zostanie dodana do wskaźnika pamięci obrazu po wyświetleniu każdej linii. Trochę to zagmatwane, ale rozpatrzmy przykład analogiczny do powyższego, w sytuacji, gdy modulo ekranu wynosi 2. Zerowa linia, tak jak powyżej, zostanie wyświetlona bazując na danych spod adresu $70000, dane dla linii następnej - pierwszej, zostaną pobrane spod adresu będącego sumą adresu linii poprzedniej ($70000), jej długości ($28) oraz ustawionej modulacji (czyli $02). W rezultacie, sumując

Czy widzicie już możliwość wykorzystania modulacji w jakimś konkretnym zastosowaniu? O tak! Przecież to jest jakby specjalnie stworzone dla miłośników pisania scrolli! W miejsce uzyskane dzięki wprowadzeniu modulacji możemy przecież wstawiać co pewien czas kolelejną literę, a całość, począwszy od miejsca oznakowanego na rysunku symbolem „KON", do adresu $70000 przesuwać w lewo. Cykliczne powtarzanie tych czynności da w rezultacie złudzenie „płynięcia" napisu czyli po prostu najzwyklejszego w świecie scroll'a. Jego prędkość przesuwu zależeć będzie od wartości o jaką będziemy przesuwać jednorazowo zdefiniowane okno w pamięci obrazu. Stosownie do tego należy również z odpowiednią częstotliwością wstawiać kolejną literę tekstu. W naszym konkretnym przypadku, gdy modulacja ustawiona jest na 2, czyli szerokość liter wynosić będzie 2*8 bitów = 16 pixeli, oraz zamierzamy scrollować z prędkością najniższą (za każdym razem jeden pixel) nową literę musimy wstawiać co 16/1=16 przesunięć. Jeżeli natomiast chcielibyśmy scrolować z

prędkością większą, dajmy na to co 2 pixele, nową literę musielibyśmy wstawiać co 16/2=8 przesunięć. Jak łatwo się domyślić scrolowanie co 3 pixele wprowadza małe zamieszanie, jednak co 4 jest znów bardziej przyjemne i wymaga wstawienia każdej nowej litery co 16/4=4 przesunięcia. Bazując na powyższych wiadomościach napiszmy teraz procedurę wstawiania w odpowiednie miejsce pamięci ekranu danej litery, oraz samego scroll'owania. Zanim jednak to uczynimy przyjmijmy następujące założenia:

*) Wszystkie litery są dwukolorowe (kolor tła i jeden właściwy) oraz mają jednakową wielkość tzn. 16 pixeli szerokości i 16 linii wysokości.

*) W związku z powyższym na ekranie widocznych może być w całości 320/16=20 liter.

*) Modulacja ekranu wynosi 2, co daje nam możliwość „ukrycia" poza ekranem jednej litery.

*) Zestaw znaków wykonany jest jako obrazek o szerokości 320 pixeli.

A oto procedura wstawiania litery:

```
move.l    #AdresLitery,$dff050
move.l    #$00070028,$dff054
move.w    #$ffff,$dff044
move.w    #$ffff,$dff046
move.w    #$09f0,$dff040
move.w    #$0000,$dff042
move.w    #38,$dff064
move.w     #40,$dff066
move.w     #[64*16]+1,$dff058
rts
```

Co wykonują poszczególne linie? Przyjrzyjmy się im dokładnie. Pierwsza z nich przesyła do rejestrów $050 i $052 (proszę zwrócić uwagę na fakt przesyłania długiego słowa), czyli do BLTAPTH oraz BLTAPTL, adres litery w naszym zestawie znaków. Zauważmy, że adres ten jest inny i charakterystyczny dla każdej litery. Rozkaz w linii drugiej ustawia przeznaczenie, czyli adres, pod który ma zostać skopiowana litera na wartość $70028. Rozkazy w linii trzeciej i czwartej ustawiają maskę w ten sposób, aby cała litera została skopiowana ($ffff and $xxxx, gdzie $xxxx jest dowolną wartością da nam zawsze w wyniku $xxxx - aktualnie interpretowane jako kształt litery). Kolejny rozkaz ustawia bity 4,5,6,7,8,11 w zerowym rejestrze kontroli. Odpowiadają one za:

4,5,6,7 : Są kodem wykonywanej operacji, w tym przypadku oznaczają zwykłe kopiowanie.

8 : Oznacza użycie kanału przeznaczenia.

11 : Oznacza użycie kanału źródła A.

Następny rozkaz kasuje bity w pierwszym rejestrze kontroli Blitter'a. Dalej ustawiamy modulację Blitter'a dla źródła A na wartość 38. Modulacja ta „mówi", że po skopiowaniu każdej linii litery, Blitter ma dodać do wskaźnika źródła wartość 38, aby trafić na początek kolejnej linii litery. W praktyce suma modulacji źródła i szerokości litery ma dać wartość równą szerokości obrazka, z którego pobieramy literę lub inną jego część (teraz też literę możemy traktować jako część obrazka składającego się z całego alfabetu). Następny rozkaz ustawia z kolei modulację dla przeznaczenia na wartość 40. Ta modulacja, analogicznie do poprzedniej, jest wartością którą doda Blitter do wskaźnika przeznaczenia po skopiowaniu każdej linii litery, w celu obliczenia adresu wstawienia kolejnej linii znaku. Wreszcie ostatni rozkaz definiuje wielkość kopiowanego bloku i oznajmia Bliter'owi, że wszystko jest przygotowane, więc może wziąć sie do pracy. Wartość 1 w operandze jest szerokością naszego znaku wyrażoną w słowach (przypominam, że słowo to dwa bajty, czyli 16 bitów odzwierciedlających 16 pixeli). Natomiast wartość 16 jest wysokością znaku. Iloczyn z liczbą 64 powoduje przesunięcie liczby 16 na wyższe, odpowiedzialne za pionową wysokość bity.

A oto procedura scroll'owania o jeden pixel w lewo. Dokładnie zostanie skomentowana za miesiąc:

```
move.l    #$70000+[17*42]-1,$dff050
move.l    #$70000+[17*42]-1,$dff054
move.w    #$ffff,$dff044
move.w    #$ffff,$dff046
move.w    #$19f0,$dff040
move.w    #$0002,$dff042
move.w    #$0000,$dff064
move.w    #$0000,$dff066
move.w    #[64*17]+21,$dff058
rts
```

Krzysztof Kobus

P.S. Kompletna procedura scroll'a wykonana wg powyższego algorytmu znajduje się na naszym drugim Kebab-Public-Domain-Dysku.

"Listów" ciąg dalszy

zapewniam Pana, że listingi w **oryginalnej** instrukcji obsługi są wydrukowane poprawnie tzn. bez błędów. Niemniej jednak niektórzy "ambitni" dostawcy hurtowi sprzętu postanowili dołączać do komputera tzw "polską instrukcję obsługi". Cieszy nas osobiście fakt, że dzięki temu rowniez nabywcy, którzy nie ukończyli studiów germanistycznych (ew anglistycznych) mogą czegoś się dowiedzieć na temat obsługi dopiero co zakupionego sprzętu. Smuci natomiast to, że w/w tłumaczenie jest oprócz paskudnego wykonania estetyczno poligraficznego (wiadomo: koszty), jest również upstrzone całą masą błędów merytorycznych. Nie znaczy to oczywiście, że mamy wspomnianą książeczkę wyrzucić od razu najbliższego kosza na śmieci. Musimy po prostu postarać się o oryginał. Na szczęście nie wszyscy dostawcy wyrzucają go z pudełka, aby zrobić miejsce dla polskiej wersji. Gdy już zdobędziemy oryginalną instrukcję, musimy odszukać wszystkie "mało istotne" szczegóły, którymi różnią się listingi przykładowych programow w j. BASIC. Może to być nawet zajmujące. Przypomina mi się znana zabawa ze środkowych stron "Przekroju" pt. "Wytęż wzrok..." i znajdź xx szczegółów... Pamiętac należy przy tym, że **KAŻDA** odnaleziona różnica jest błędem, który najprawdopodobniej nie pozwoli na poprawne działanie przykładu.

Sprawa druga

Programy, o które Pan pyta istnieją na rynku Public Domain dla C-64 i umożliwiają uzyskanie rozmaitych "bajeranckich" efektów. Mam tu na myśli wszelkiej maści Demo-, czy Intro-Maker y. Wspólną ich wadą jest to, że nie powstały w Polsce i w związku z tym ktoś "zapomniał" zainstalować w nich polskie litery (jak zwykle powraca problem ogonka). Co w związku z tym mozemy zrobić? Postaramy się w jednym z najbliższych KEBAB'ów zamieścić tego typu program umożliwiający przynajmniej wykonanie podstawowych efektów z użyciem polskich liter. Jeżeli zainteresowanie (odzew Czytelników) będzie wystarczająco duże, być może zdecydujemy się również na wydanie tego typu programu na poziomie pro.

Szanowna redakcjo!
Z radością przyjąłem fakt, że od numeru 4/92 pan Krzysztof Moroń rozpoczął cykl artykułów na temat AMOS'a...
L. Jakubiak

... Dlaczego przestaliście opisywać AMOS?... ... czy nie moglibyście prowadzić kursu w tym języku podobnie jak prowadzicie kurs języka maszynowego?
M. Lisman

c d. na str 35



Gobliiins,

czyli sporo niespodzianek.

Wszystkim, którzy lubią trochę pogłówkować w czasie grania z pewnością spodoba się gra pod tytułem „Gobliiins". Z gruntu prosta fabuła, dobra grafika, a przede wszystkim kapitalne pomysły, duże poczucie humoru i zagadki, zagadki, i jeszcze raz zagadki. Przejdźmy jednak do bohaterów naszej opowieści. Są to trzy małe, czerwono ubrane stworki o różnych zdolnościach i umiejętnościach. MERLIN, to ten łysawy z siwą bródką i żółtymi oczami, gdy pracuje. Ten facio naprawdę zna się na czarach. Rozumiem, wyczarować z jabłka większe jabłko, ale żeby z kreta ... (patrz etap 13). Zmyślnej głowie przyda się ktoś, komu Bozia nie dała „pod rogami", ale za to nie poskąpiła na mięśniach. FIZOL przyda się nam wiele razy podczas naszej misji. Ostatnim członkiem wyprawy jest KIESZONA. Tylko on ma możliwość podnoszenia (zostawiania) przedmiotów, a także ich używania. Ruch postaci odbywa się przy wskaźniku w kształcie strzałki, zbieranie/zostawianie przy otwartej dłoni, cios/czarowanie/używanie obrazuje zaciśnięta pięść. Zmianę wskaźnika uzyskuje się prawym przyciskiem myszy. Lewy służy do wykonania polecenia. Gra składa się z 22 etapów, każda ma swój kod, który umożliwia wejście na dowolny poziom gry. Opisywana wersja Gobliiins'ów miała ustawioną klawiaturę Azerty. Tym, którzy mają kłopoty ze znalezieniem potrzebnych do kodów liter przyda się prawdopodobnie mała ściągawka. Tak więc zamienione są tu litery W z Z, A z Q, litera M znajduje się pod klawiszem O umlaut (niemiecka klawiatura). Zacznijmy więc od początku (nie opuszczajcie intra, jest świetne!). Już po krótkiej chwili możemy się zorientować, że ktoś bardzo nie lubi naszego Króla. Z dziką rozkoszą zapewne dźga magiczny fetysz igłami, traktuje go młotkiem, łechce piórkiem i wymachuje pająkiem. Biedny Król odchodzi od zmysłów i pozosaje mu tylko nadzieja w jego wiernych sługach. Trzech naszych przyjaciół wyrusza niezwłocznie na poszukiwanie sprawcy tego nieszczęścia.

Etap 1 - zaczyna się przed chatą Czarnoksiężnika. Przyjrzyj się uważnie, a zauważysz poruszający się na bramie róg. Podejdź FIZOLEM do prawego filaru i uderz w niego pięścią. Róg spadnie na ziemię. Teraz do akcji wkracza KIESZONA, podnosi róg i w niego dmie (używa). Z drzewa spada gałązka. Zaczaruj ją MERLINEM. Zamieni się ona w kilof. KIESZONA wchodzi w posiadanie tego narzędzia i w ten sposób kończymy ten etap (kliknij na okno GO!).

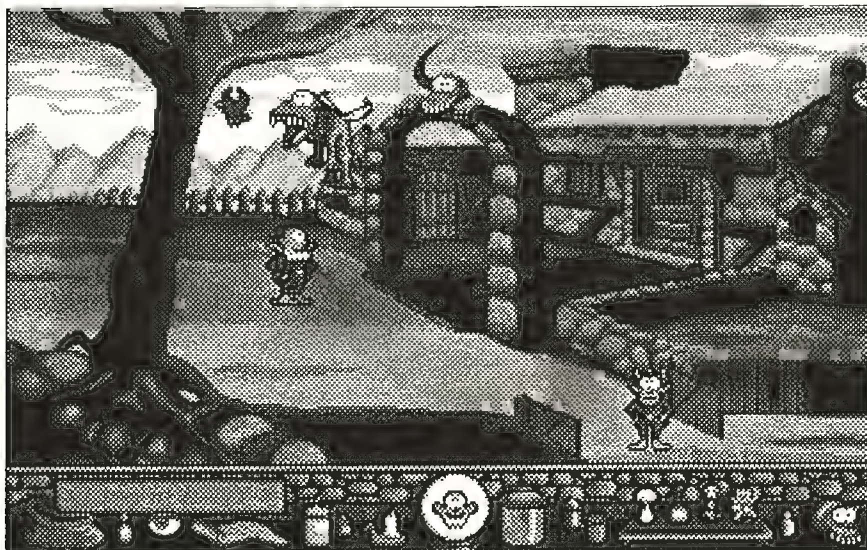
Etap 2 - (VQVQFDE). MERLIN czaruje jabłka. Po chwili staną się nazwijmy to dorodne. Teraz FIZOL pokaże co potrafi, uderza pięścią w, licząc od lewej, drugie i czwarte jabłko. Na resztę szkoda jego wysiłku, są robaczywe. Teraz KIESZONA przenosi opadłe jabłka nad przepaść i tam je używa. Wraca po pozostawiony kilof i używa go w grocie po przeciwnej stronie. Tak dorabia się w końcu diamentu.

Etap 3 - (ICIGCAA). Niech FIZOL jak najładniej potrafi zapuka do drzwi wejściowych...

Etap 4 - (ECPQPCC). KIESZONA musi się rozstać na moment z diamentem i zabrać pierwszy od lewej pojemnik (pot 1) ze stolika z drabinką. Następnie używa go na prawym kwiatku. Z pojemnika wyleci owad i mamy trzy sekundy na przegląd uzębienia roślinki. W tym czasie MERLIN czaruje lewy kwiat. Ten z kolei wyciąga się nieco. Wystarczy to jednak FIZOLOWI, który wdrapie się po nim i wyładuje nadmiar energii na książce w prawym rogu. KIESZONA wraca po diament i po książce wdrapuje się na biurko. Otwarta dłoń Czarnoksiężnika niedwuznacznie daje do



Nasz kochany król nie ma chwili spokoju...



Etap 11 - bestia odlatuje wraz z ptaszkiem...

zrozumienia co KIESZONA ma zrobić z diamentem. Jeszcze tylko krótka wymiana zdań, z której wynika, że musimy udać się pod ziemię, by odszukać dla czarnoksiężnika trzy rzeczy: Airain's Mushroom, Archnide's Elixir i Bald Plant.

Etap 5 - (FTWKFEN). MERLIN schodzi na sam dół, gdzie czaruje wystający z ziemi tuż obok garbusa pręt. Pręt, jak to pręt, oczywiście się rozrasta. FIZOL, korzystając z okazji, „przykłada" pięść do prawego oka posągu. I oto mamy zgrabną windę. KIESZONA pakuje się na języko-windę, a MERLIN w tym czasie czaruje wygięty kształt nad sarkofagiem z lewej strony u góry. Teraz szybkim krokiem musi zbiec do KIESZONY na język. FIZOL raz jeszcze sprawdza dopasowanie oka posągu do swojej dłoni i cała ferajna jest już na górze. Po chwili z sarkofagu wyjdzie monstrum, które przestraszy garbusa, roześmieje się urodziwie i wróci do siebie. Ostatni raz FIZOL uruchomi windę i KIESZONA podniesie pozostawiony przez garbusa grzybek. A grzybek jest to nie byle jaki, bo dokładnie ten, którego szuka Czarnoksiężnik.

Etap 6 - (HQWFTFW). Pająki bywają złośliwe, dlatego lepiej będzie jak MERLIN profilaktycznie rzuci dwa razy urok na paskudę blokującą most. FIZOL wspina się na najwyższy poziom i uderza w ostatnią od lewej, krótką nić pajęczą. Pajączek podsunie się do góry na tyle, by KIESZONA dotarł do pistoletu. Temu to dać! Najpierw strzela do pająka blokującego wyjście z jaskini. Ten również podciągnie się do góry. Następną ofiarą jest śpiący po lewej stronie zwierzak. Tego nie dość, że uszkodzi, to jeszcze obrabuje. Skradzioną poduszkę kładzie na moście pod pająkiem. MERLIN czaruje nieboraka jeszcze raz i ten gubi w końcu poszukiwany przez Czarnoksiężnika elixir.

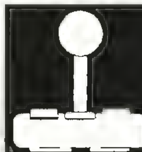
Etap 7 - (DWNDGBW). MERLIN zaczyna od zaczarowania torebki z nasionami na drzewie. KIESZONA, facet z natury praktyczny, używa ją na zaoranym polu tuż przy strachu na wróble. FIZOL oczywiście i strachowi nie przepuści, okładając go po brzuchu w czasie prób wyjadania nasion przez ptaki. Ale roślin coś nie widać. Długo się nie zastanawiając MERLIN czaruje drugą chmurę po prawej stronie drzewa. Deszcz przyspiesza wzrost kwiatu, ostatniej brakującej dla Czarnoksiężnika zdobyczy. Pojawia się on zresztą natychmiast po odbiór swoich rzeczy.

Etap 8 - (JCJCJHM). Niestety okazuje się, że ktoś nas tutaj wykiwał. Lądujemy w lochu z bardzo miłym towarzystwem. MERLIN czaruje szkielet nieopodal drabinki. Szkielet z tej wielkiej radości gubi piszczel (jeden w tą, jeden w tamtą...), który nasz czarodziej zamienia we flet. Ten z kolei, użyty przez KIESZONĘ na śpiącej kobrze, powoduje tradycyjne rozciągnięcie delikwentki na kształt drabiny. Po niej FIZOL dostaje się na górę i uderza w kupę kamieni po prawej stronie ekranu. Teraz wystarczy tylko wyekspediować MERLINA i KIESZONE z deski, na którą FIZOL będzie spuszczał kamienie.



Etap 9 - (ICVGCGT). KIESZONA musi podejść do mięsa (i użyć go na miejscu) jak najmniej zbliżając się do krokodyla. Bardzo to oryginalny „zły pies", ale da się go obejść.

Etap 10 - (LQPCUJV). KIESZONA zaspokaja apetyt mieszkańca najbliższej dziupli. Następnie bierze „wietrzną skarpetkę" (bardzo swobodne tłumaczenie zważywszy, że chodzi o urządzenie do wizualnego pomiaru siły i kierunku wiatru), po czym robi sobie przerwę na lunch. MERLIN zajmuje się w tym czasie gałązką z powiewającymi listkami. Zgadnijcie co robi gałązka. Otóż gałązka się jak zwykle wydłuża. MERLIN grzeje na jej skraj. FIZOL natomiast wdrapawszy się na największy kamień głuszy odstający kawałek gałęzi. Uporawszy się z MERLINEM, to samo robi z KIESZONĄ. Czary okazują się wielce przydatne przy wyciąganiu z gałęzi korka. KIESZONA korkuje następnie dziuplę w grubej gałęzi i wraca po „wietrzną skarpetkę", która tym razem posłuży jako siatka na ptaki. FIZOL uderza pięścią w dziuplę z ptaszkiem. Ptaszyna spłoszona takim traktowaniem wpada prosto w siatkę KIESZONY.



Etap 11 - (HNWVGKB). Znowu bardzo ostrożnie KIESZONA podchodzi od przodu do krokodyla i „pokazuje mu" ptaszka (zdecydo-

wanie chodzi tu o tego przed chwilą złapanego). Gadzina wydaje się być wyraźnie zauroczona tym niecodziennym widokiem. MERLIN podchodząc również od przodu czaruje krokodyla. Romantyczna dusza tego ostatniego odlatuje wraz z ptaszkiem. FIZOL z sobie wrodzoną delikatnością zapuka w drzwi dobudówki.

Etap 12 - (FTQKVLE). KIESZONA znajduje zabawkę (piłkę z patyczkiem), którą następnie prezentuje kościom lewej dłoni szkieletu. Następnie wyjmuje z kałamarza pióro i łaskocze nim stopę kościotrupa. Z otwartej czaszki wypada kluczyk, który KIESZONA podaje tajemniczemu więźniowi. MERLIN w międzyczasie czaruje pozostawione pióro i przerabia go na packę na muchy. KIESZONA natychmiast próbuje wynalazek na przygodnej pszczole. Dymek ze zwłok informuje nas, że osa wydała ostatnie tchnienie. Czary MERLINA separują z jej trupa żądło. KIESZONA natychmiast używa go na obrazie Czarnoksiężnika. Z szafki opodal wypada fetysz, z pomocą którego zły czarodziej zadawał męki Królowi. KIESZONA zabiera lalkę i elixir przy szkielecie.

Etap 13 - (DCPLQMH). MERLIN czaruje najbliższy wystający korzeń. Podnosząc się ukazuję trąbkę do wabienia ptactwa. KIESZONA bierze ją i używa po wspięciu się na poziom gniazda. Na gnieździe wkrótce zasiądzie ptak. Z jaja wylęgną się jedynie nogi, a po interwencji MERLINA skrzydła. Wystarczy to jednak by przenieść maga na drugą stronę. Ukryty za murem MERLIN czaruje leżący na drodze klakson. KIESZONA bierze z powrotem elixir i używa go przy końcu murka. Niewidzialny przechodzi na drugą stronę. FIZOL również podąża do skraju muru. MERLIN czaruje znęconego zapachem kreta, który zamienia się w ruchomy, jak żywy akt kobiecy w 4D! No niewątpliwie przy tego rodzaju odwróceniu uwagi Czarnoksiężnika FIZOL może trzy razy przemaszerować tam i z powrotem.

Etap 14 - (EWDGPNL). MERLIN czaruje najbliższy długi kamień. KIESZONA bierze kij i po utworzonych schodkach włazi na duży kamień. Wsadza kij w dziurkę w dolnej prawej części i otrzymuje konewkę. Podlewa teraz drugą, trzecią i ostatnią (od prawej strony patrząc) roślinkę. MERLIN czaruje interesujące nas byliny. FIZOL uderza w marchew z kluczykiem, który KIESZONA używa na sąsiedniej roślinie. Trzecia marchewka to już sama przyjemność.

Etap 15 - (TCNGTOV). W tym etapie trzeba mieć nie lada kondycję, gdyż biegania jest tu sporo. FIZOL uderza w armatę tak, by celowała do góry. Następnie uderza w kule armatnie poniżej. KIESZONA bierze kulę armatnią i pakuje ją do lufy, goni po zapałki na górnym poziomie. FIZOL znowu uderza armatę tak, by patrzała do góry, a KIESZONA używa na niej zapałek. Po huku ze sklepienia wypadnie marchewka. KIESZONA używa zapałek w kuchni i kursem powrotnym zabiera marchew, którą wrzuca do lufy. FIZOL uderza armatę tak, by celowała na wprost. KIESZONA przynosi zapałki i odpala działo. Marchewka trafia do garnka, a jej zapach budzi śpiocha. Całą zabawę powtarzamy jeszcze raz z nową kulą i armatką wycelowaną do góry. Nową marchewkę czaruje MERLIN. KIESZONA używa tuby do nagłośnienia na przygłuchym gospodarzu. Po krótkiej rozmowie KIESZONA bierze pałkę, którą używa na wiszącym powyżej gongu. Zabieramy wahadło i heja!

Etap 16 - (TCVQRPM). KIESZONA bierze kamień i kładzie go na miejscu oznaczonym krzyżykiem. MERLIN czaruje go dwa razy. To samo czeka pierwszą z lewej palmę. FIZOL uderza powstały kilof, KIESZONA używa go kilka razy na środku ekranu.

Etap 17 - (IQDNKQO). FIZOL uderza w stos drewna po prawej stronie. KIESZONA używa wypadnięty pieniek na sidłach przy mostku. MERLIN czaruje woreczek na poziomie smoka (nie zapomnij się zaraz potem wycofać). KIESZONA zostawia polano idzie po woreczek. MERLIN przemienia pozostawione polano na dezodorant. KIESZONA używa woreczka na górnym poziomie i zwabia stopę. Dezodorantem uspokaja ją bardzo i stawia smokowi na podeście. Świeżo wypieczoną oddaje pułapce na moście. Teraz spokojnie zabiera miecz i stawia go smokowi na podpałkę. Płonący zabiera i już mu niewiele do szczęścia potrzeba.

Etap 18 - (KKKPURE). KIESZONA używa płonący miecz na okrągłej tarczy na brzuchu posągu, wchodzi na dłoń z prawej strony ekranu, a następnie na drugą. Bierze klucz i używa go na głowie bóstwa. Teraz wszyscy po kolei znikają na dłoni po prawej.

Etap 19 - (NGOGKSP). KIESZONA bierze mydło i używa go na pisarzu. FIZOL uderza w kiść bananów. KIESZONA bierze opadnięty banan i używa go na pisarzu. Z górnej półki zabiera sztuczny nos i także używa go na delikwencie. Tak zachęcony do pracy szybko kończy swoje dzieło - Księgę Czarów. MERLIN czaruje bramę wyjściową w wieży.

Etap 20 - (NNGWTTO). MERLIN czaruje drobny koniuszek w ziemi tuż przy sarkofagu. FIZOL uderza w powstałą dźwignię. MERLIN czaruje korek w uchu olbrzyma, KIESZONA używa Księgę Czarów na tym uchu. W lewej wieży znajduje rybę i zostawia ją w miejscu, gdzie leży miska. Podstawia miskę pod oko olbrzyma z lewej strony. Powtórnie używa książki na uchu. Miska napełnia się łzami i jest gotowa do użycia jej na potworku znęconym rybą.

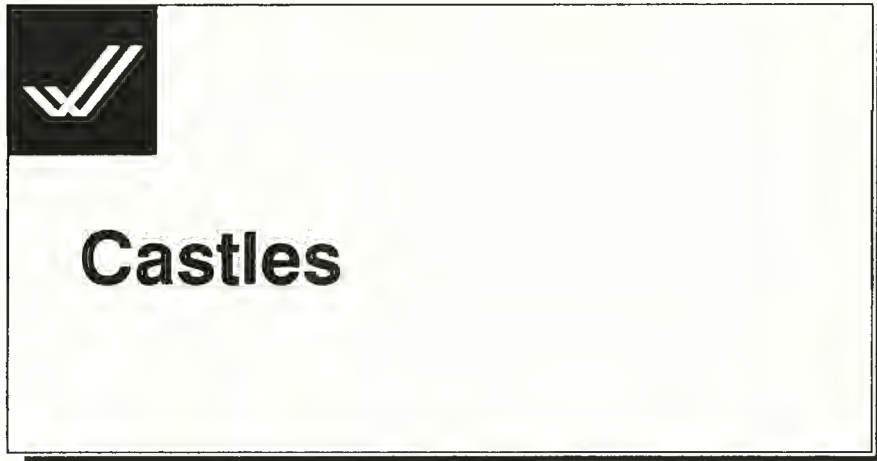
Etap 21 - (LGWFGUS). KIESZONA strzela z procy do bananów. FIZOL uderza w dźwignię i wszyscy wsiadają na rekina.

Etap 22 - (TQNGFVC). MERLIN czaruje pierwszy od lewej koniu-

szek nogi szkieletu ptaka. KIESZONA wchodzi po linie i odstrzeliwuje z procy linę. Używa ją następnie na kolcu wystającym z górnego skrzydła szkieletu. Bierze worek i kładzie go w rogu ekranu, nad rekinem. Następnie z procy trafia Czarnoksiężnika. MERLIN czaruje go i dziwne rzeczy dzieją się z nim w kotle. FIZOL wchodzi po linie i znowu folguje sobie swojemu hobby, powodując kolejną przemianę Czarnoksiężnika. KIESZONA otwiera ogień swojej procy i uwalnia FIZOLA. MERLIN po raz ostatni czaruje złego Czarnoksiężnika, który w postaci małych pajączków kończy karierę w worku KIESZONY.

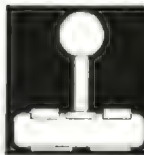
Król znowu jest wolny i zdrowy, nasi bohaterowie cało i zdrowo powrócili do domu, odtąd żyli długo i szczęśliwie...





Castles

Bądź pozdrowiony Panie nasz i władco!!! Niech Bóg przyświeca twoim przedsięwzięciom ... Tak zaczyna się twoja nowa przygoda. Zaszczyty, władza, własna armia, twierdze, polityka ... Wszystko to znajdziesz w grze pod tytułem „Castles". A zabawa jest przednia. Twoim głównym zadaniem jest budowa i obrona własnoręcznie zaprojektowanych zamków. Jako niepodzielny władca musisz też, w międzyczasie, godzić zazwyczaj sprzeczne interesy swoich podwładnych. Zanim zaczniemy grę, musimy najpierw zadecydować o kilku ważnych dla jej przebiegu rzeczach. Tak więc, wybieramy prędkość (steady, swift, laboured), ilość zamków (1-8), stopień trudności (peasant, duke, prince, king), wiadomości z królestwa, wreszcie świat w jakim przyjdzie nam egzystować (real, fantasy). Po wpisaniu naszego imienia i nazwy przyszłej budowli, możemy zabrać się do pracy. Na początku pokaże nam się obraz, który ułatwi nam strategiczną lokalizację naszego zamczyska. Wskazane jest więc miejsce, do którego dostęp jest jak najbardziej ograniczony, np. wodą. Bliskość wody umożliwia także przyszłą budowę fosy. Należy również pamiętać, by przedpole, z którego spodziewamy się ataków było jak najbardziej odsłonięte. Ewentualne zarośla, skały znacznie utrudnią nam późniejsze odpieranie ataków. Praktycznie jest także, na dobry początek, rozpocząć budowę tylko kilku elementów, jedynie z grubsza planując całość naszego dzieła. Z praktyki tej gry wynika, że ewentualne ataki następują co najwyżej z dwóch kierunków i to przeważnie sąsiednich np.: S i SW. Po pierwszym ataku wiemy już więc, z której strony twierdza musi być szczególnie wzmocniona. Pozwoli nam to także umiejscowić delikatną i bardzo pracochłonną bramę. Przesuw ekranu odbywa się po najechaniu kursorem na ramkę obrazu odpowiedniego kierunku i przyciśnięciu lewego przycisku myszy. Jeżeli już dokonałeś wyboru kliknij RMB. Z lewej strony ukaże się główne menu, a w nim ikony: Design, Labour, Taxes, Military, Food, Options. Poniżej mamy aktualną datę, dane o grającym, skarbcu, ilości ukończonych i nieukończonych partiach budowli oraz „środkach inwestycyjnych". Pierwszą niezbędną czynnością jest wynajęcie robotników. Wybieramy więc ikonkę Labour i przyglądamy się nowemu menu. Mamy tu całą listę zawodów: kopacz, cieśla, murarz, kamieniarz, taczkarz i zwykły robotnik. „Hire" pozwoli nam podnieść aktualny stan zatrudnionych. „Fire" dokona „zwolnienia grupowego". U dołu mamy dane o liczbie robotników, ich płacach i stanie skarbca. Płace możemy zmodyfikować klikając na ikonie „Wages". Na liście płac dodatkowo znajdują sie żołnierze i łucznicy. Większe płace umożliwiają zwiększenie maksymalnej liczby zatrudnionych, szybszy najem, a także powodują ich lepszą pracę. W czasie napływu robotników my zajmiemy się nadawaniem kształtów naszej twierdzy. Z głównego menu wybieramy Design. Mamy tu do wyboru dwa rodzaje baszt, mury, bramę oraz „buldożer". Poniżej znajdują się bardzo ważne informacje o każdej budowanej partii. Tak więc od góry: planowana wysokość, ilość pracujących tu robotników, ilość dni do zakończenia budowy, aktualny stan (no workers, waiting, building, unstable, complete), otwory strzeleckie i do lania wrzątku (w przypadku murów także ich grubość), środki, ilość najętych robotników „bez przydziału" i w końcu aktualna wysokość. W zależności od fantazji możemy położyć fundamenty całego zamku, a następnie stopniowo wznosić poszczególne elementy. Nigdy nie buduj samego muru niepodpartego basztami (chociażby w budowie) z obu stron !!! Przeważnie kończy





Nasz zamek może wyglądać np. tak...

się to komunikatem „unstable" i uroczym nagrobkiem dla zatrudnionych tam robotników (zawaleniem). Teraz możemy już postarać się o armię która zatroszczy się o obronę budowli (główne menu, Military). Strzałkami ustalamy przyszłą liczebność armii. Maksymalna liczba żołnierzy zależy od ilości zaplanowanych elementów twierdzy. Większa budowla umożliwia kwaterunek większej armii. Bardzo ważny jest tu wskaźnik Rating (mediocre, poor, good, excelent, elite). Oddziały „liche" lub „słabe" będą w czasie bitew padać jak muchy. Mamy tutaj także opcję Moat. Po jej użyciu wokół naszego zamku pojawią się kopacze, którzy w ekspresowym tempie wykopią zgrabną fosę. Ikona Taxes daje nam pełną kontrolę nad finansami państwa. U góry możemy ustalić poziom rocznego opodatkowania. Generous to roczne dochody na poziomie 2000 funtów, Normal 2500, Opressive 3000 i Tyranic 3500. Poniżej możemy każdego miesiąca uchwalić extra podatek (Levy) w przypadku przejściowych kłopotów finansowych. Jego wysokość ustalamy strzałkami, wypełniony kwadracik oznacza Twoją zgodę na ściągnięcie podatku w tym miesiącu. Na samej górze możemy uzyskać dokładne sprawozdanie o stanie budżetu państwa (Treasurer). Lista obejmuje wykaz miesięcznie ponoszonych kosztów na siły zbrojne, „klasę robotniczą" i pozostałe. Poniżej mamy wykaz rocznych dochodów ze wszystkich zamków. W zależności od stanu skarbca skarbnik ostrzega nas lub doradza kupno najpotrzebniejszych dóbr. Wybierając Food z głównego menu, możemy zaopatrzyć nasz zamek w zapas żywności. Maksymalnie w ciągu jednego miesiąca możemy nabyć 150 jednostek. Cena jednej jednostki zmienia się oczywiście w ciągu roku od 3 do 1 funtów. Na przednówku jest oczywiście najwyższa, dlatego praktycznie jest kupować w czasie ostatnich trzech miesięcy. Zbagatelizowanie zapasów żywności może spowodować znaczny ubytek w armii i wśród robotników w czasie długiej zimy. Szczególnie ważna jest nasza spiżarnia, gdy nasz przeciwnik wpadnie na pomysł oblegania zamku. Stopnieją wtedy albo twoje zapasy, albo szeregi twojej armii. Od czasu do czasu głodne szczury lubią wpaść na „małe conieco" (niech mi Kubuś Puchatek wybaczy...), nie zapomnij więc często kontrolować stanu magazynów. Tak więc, zostało nam już tylko porozwodzić się chwilę nad ostatnią ikoną, Options. Z grubsza jej menu wygląda na standardowe, a więc Load, Save, Quit, Speed, Main. Jest tu jednak bardzo przydatne nowum, Counsel. Po kliknięciu na tej ikonce przed nasze oblicze stawi się Sir Richard z Westhampton, jeden z najświetniejszych rycerzy Realm'u. Z bliżej nieznanych źródeł posiada on zawsze aktualne informacje o tym, co myślą o tobie możnowładcy, Kościół oraz prości współziomkowie. Sir Richard z Westhampton zawsze gotów jest udzielić Ci wszelkich porad.

Po tych „technicznych" szczegółach przejdzmy do przebiegu gry. Zaraz po położeniu fundamentów i najęciu robotników, na placu budowy zaczynają się kręcić przeróżne postacie. Twoja twierdza rośnie w oczach. Po krótkim czasie murarze stoją już na rusztowaniach podciągając sobie wiadra z materiałem na linach, wokoł kręcą się taczkarze, cieśle piłują drewno na przyszłym dziedzińcu. Nad wszystkim czuwa zarządca na koniu. Całość wygląda szalenie sympatycznie i skutecznie zabija czas oczekiwania na inne doniosłe wydarzenia. W międzyczasie Ty zajmujesz się „rządzeniem". Co chwila przychodzą do Ciebie różni ludzie z prośbą o radę, żądaniem, narzekaniem itp. A problemy są tu najróżniejsze. Pół biedy gdy jest to twój wierny rycerz z prośbą o nadanie imienia swojej córce, Twój syn po bójce w karczmie, czy Biskup na partyjce szachów (za którymi nie przepadasz). Zdarza się jednak często, że gdzieś rozrabia jakiś wyjęty spod prawa banita, ktoś oferuje Ci przymierze, inny prosi o wsparcie w walucie lub wojsku, Twój dawno wygnany brat chce powrócić do łask, Kościół wiecznie widzi we wszystkim szatana którego z pomocą twej armii ma ochotę okiełznać itd, itp. Większość problemów jest tak sformowana, że musisz dokonać wyboru pomiędzy interesami różnych grup społecznych. Balansowanie na tej cienkiej niteczce na prawdę nie jest proste. Gdybyś miał z tym problem, pamiętaj o Sir Richard'zie z Westhampton. On podpowie Ci, kto ewentualnie jest niezadowolony z twoich rządów. To jednak nie wszystkie problemy jakie na Ciebie czekają. Od czasu do czasu twoją

budowę przerywają potyczki z armią wroga. Na początku gry oddziały te nie są zbyt silne. W przypadku wytropienia wroga ekran automatycznie przełącza się i możemy rozpocząć rozmieszczanie naszych oddziałów (zwróć uwagę na kierunek z jakiego nadciąga nieprzyjaciel). Łucznicy najlepiej komponują się na basztach i murach. Piechotę warto rozmieścić pod murami twierdzy. Jak tylko uporamy się z tym problemem, rozpoczyna się bitwa. Twoja rola ogranicza się do wyboru celu dla poszczególnych jednostek. Kliknij więc na swojego żołnierza (pojawi się żółty kwadrat), a następnie na oddział wroga (czerwony). Od tej pory będzie to już nierozłączna para aż do śmierci jednego z nich. Taktyka jest już twoim pomysłem. Lepiej jest chyba rozegrać bitwę nie dopuszczając wroga pod mury. Tam może ich czekać co prawda bardzo gorąca kąpiel (wrzątek z otworów w murze), ale uszkodzenie lub zniszczenie części zamku może być przykre w skutkach. Jego odbudowa zabierze dużo czasu i pieniędzy. W miarę rozwoju gry nasz przeciwnik atakuje coraz skuteczniej. Najgroźniejsze są awantury z możnowładcami, których posłańców właśnie odesłaliśmy z kwitkiem. Ich armie atakują przeważnie w dwóch rzutach. Przestaje tu działać zasada, że ataki następują z sąsiednich kierunków. Przeciwnicy ciągną także ze sobą specjalne machiny wojenne do robienia wyłomów w murach. Fosa jest tu jednak wystarczającą bronią. Po zakończonej bitwie uzupełniamy ubytki w budowli i armii. W takich warunkach próbujemy dokończyć nasz zamek. Sir Richard z Westhampton powie Ci ile części trzeba jeszcze wybudować by zebrać większe podatki (nie jest to niezbędne dla skończenia gry). Kiedy wszystkie części (twierdze w przypadku opcji gry z ośmioma zamkami) są już gotowe, czeka Cię jeszcze decydująca bitwa. Musisz obronić swój dopiero co zrealizowany pomysł. Po tym ostatecznym zwycięstwie komputer podsumuje twoje wysiłki, raz jeszcze objawią się opinie możnych, Kościoła, prostych obywateli. Jeszcze tylko „Dziękuję za grę..." i to już koniec. Powodzenia!

Mc Greg

Ogłoszenia drobne...

Sprzedam C-64, joystick, magnetofon, moduł, 600 programów, gwarancja, pokrywa
Andrzej Tomulewicz
ul. Stroma 35/8
15-662 Białystok
tel. 613362

Kupię cartridge "Simon's Basic". Oferty listowne:
Adam Nagórka
ul. Kilińskiego 10/29
28-200 Staszów

Nawiążę kontakt z osobami posiadającymi A500 i programującymi w AMOSie.
Maciej Lisman
ul. Jugosłowiańska 37/F/9
Stargard Szcz.

AMIGA - wymienię programy
Dariusz Sitek
ZWM 24/83
42-200 Częstochowa

Pilnie sprzedam C-64 (stan idealny) + peryferia. (info: koperta + znaczek)
Krzysztof Piętak
ul. Ludowa 15
32-547 Jaworzno

Sprzedam "Bazę danych" dla C-64 (dysk + instrukcja + dodatkowy program) cena 28000zł.
Dominik Latusek; Os. Kosmonautów 23/4; 61-692 Poznań

Sprzedam A500 Action Replay Mk II (1,3 mln zł.)
Mariusz Piechówka
Box-98
74-100 Gryfino

Sprzedam: drukarkę LC-200 Color (wbudowane polskie znaki). Cena 4 mln zł.
Marcin Balewicz
ul. Odrzańska 10/103
30-408 Kraków

Poszukujemy do sprzedaży legalnego oprogramowania peryferii IBM, Amiga, Commodore "Komis" S.C. Studio Komputerowe J&K
ul. Kolejowa 1 lp.
63-800 Gostyń

Sprzedam Amigę 500, modulator TV, Joystick, dyskietki. Orientacyjna cena 5.500.000 zł.
Sławomir Krzych
ul. Kard. Wyszyńskiego 48/81
Konin
tel.42-18-15

Sprzedam sampler mono (40KHz) + dysk 3,5" + kabel + instrukcja + 1 rok gwarancji. 400.000,-
Piotr Laszczyk
Szeligówka 976
34-511 Kościelisko

Zamienię motorower JAWA i gry TV na C64/128. Listy na adres:
Bartłomiej Ryłko
Istebna 500
43-460 Wisła
woj. Bielskie

Wymienię oprogramowanie na C-64 i / lub Amigę. Szukam mapy pamięci do C-64.
Piotr Pacyna
ul. Poznańska 3
64-200 Wolsztyn

Sprzedam Amigę 500 (v1.3; 1MB) + liter. + modulator + 2 joysticki + 460 dyskietek z grami. Cena 12 mln.
Robert Lipiec
ul. J. Śliwki 4/5
44-100 Gliwice

Drukarkę Star LC-20 na gwarancji tanio sprzedam.
Olgierd Grunau
ul. Limanowskiego 3/1
71-664 Szczecin

Najlepsze gry, dema, użytki na Amigę. Katalog: koperta + znaczek
Radosław Twardzik
ul. Przemysłowa 31
22-100 Chełm

Sprzedam Amigę 500, 1MB, zewnętrzna stacja 3A1, monitor 1084S, 320 dysków + 2 x disk box
Rafał Roszak
ul. Lubomirskiego 1/7
71-505 Szczecin

Sprzedam C-64 z magnetofonem, final III, black box v.2, ok. 600 programów, literatura (ok. 2500 tys. zł.)
Jarosław Michalski
ul. Wolności 3/4
22-100 Chełm

Wymiana obszernej biblioteki oprogramowania, literatury, doświadczeń, C=64 C=128, stacja dysków, drukarki
Commodore Club
ul. Chopina 7
57-320 Polanica

AMIGA - programy public domain, gry, użytki (koperta + znaczek)
Tomasz Chojnacki
ul. Powstańców Śląskich 57
32-300 Olkusz

Sprzedam C64II (gwarancja), 1541 II, Final III, joystick, dyskietki z programami. 4 mln zł.
Piotr Domeradzki
ul. Parzęczewska 30/11
95-100 Zgierz

Sprzedam GVP Impact Vision 24 z adapterem do A2000, G-Force 030 "combo" 50 MHz/4MB RAM z koprocesorem + Quantum LPS105 MB, Supra RAM 500RX, Digi Tiger II, Multivision 2000/500, Action Replay Mk III do A500 i A2000. Tylko poważne oferty. Kontakt listowny:
Silver Dream!'s
ul. Wojciechowskiego 28
71-476 Szczecin 41

Listing nr 1

```
0 REM BAZA DANYCH 64 (W) P.SOLTYSINSKI
1 REM (C) 1992 KEBAB LTD.
2 :
3 X=8:IFPEEK(186)=1THENX=1
4 POKE50000,X:GOSUB 5:GOTO 20
5 X$=CHR$(18):Y$=CHR$(146):Z$=CHR$(145)
6 D$="(TASMA)":IF PEEK(50000)=8 THEN D$="(DYSK)"
10 L$="":FOR R=1TO40:L$=L$+" ":NEXT:PRINTCHR$(5):RETURN
20 PRINTCHR$(147);L$;TAB(53);"BAZA DANYCH 64":PRINTTAB(13);"(C) 1992 KEBAB"
30 PRINTL$
40 PRINTTAB(17);X$;" MENU ":PRINT:PRINTTAB(3)X$;" 1 ";Y$;" - ZAKLADANIE BAZY"
50 PRINT:PRINTTAB(3);X$;" 2 ";Y$;" - ODCZYT BAZY ";D$:PRINT
60 PRINTTAB(3);X$;" 3 ";Y$;" - ZAPIS BAZY ";D$:PRINT
70 PRINTTAB(3);X$;" 4 ";Y$;" - OPERACJE NA ZAPISANYCH DANYCH":PRINT
80 PRINTTAB(3);X$;" 5 ";Y$;" - SORTOWANIE DANYCH W/G KLUCZA":PRINT
90 PRINTL$:PRINT" NAZWA BAZY: ";NB$:PRINT:PRINT" ILOSC DOKONANYCH ZAPISOW:";IL
100 PRINT:PRINTL$;SPC(4);"WYBIERZ OPCJE KLAWISZEM OD 1 DO 5";
110 GETA$:IFA$<"1" OR A$>"5" GOTO 110
120 IF NB$<>"" THEN 140
130 IF A$>"2" THENPRINTCHR$(147);"BRAK DANYCH!":FORT=0TO1200:NEXT:RUN
140 ON VAL(A$) GOTO 1000,4000,3000,2000,5000
800 GOSUB 900:IF A$="N"THEN2101
810 FOR T=K TO IL:FOR R=1 TO KL:B$(K,R)=B$(K+1,R):NEXT:NEXT
820 FOR R=1 TO KL:B$(IL,R)="":NEXT:IL=IL-1:IF IL=0 THEN 20
830 IF K>IL THEN K=IL
840 GOTO 2101
850 GOSUB 2110:PRINT"PODAJ NUMER POZYCJI DO MODYFIKACJI:"
855 PRINT "1 -";KL;":";:POKE19,16:INPUT P%:POKE19,0:IF P%<1 OR P%>KL THEN850
860 PRINT:PRINT"PODAJ NOWA ZAWARTOSC:":PRINT:PRINTKL$(P%);":";
865 POKE19,16:INPUT B$(K,P%):POKE19,0:GOTO 2101
900 GETA$:IF A$<>"T" AND A$<>"N" THEN 900
910 RETURN
1000 PRINTCHR$(147)
1010 IF NB$=""THEN 1050
1020 PRINT"SKASOWAC AKTUALNE DANE (T/N)?"
1030 GOSUB 900
1040 IF A$="N"THEN20
1050 CLR:GOSUB5:PRINTCHR$(147);L$;SPC(9);"ZAKLADANIE NOWEJ BAZY":PRINTL$
1060 POKE19,16:PRINT"PODAJ NAZWE DLA NOWEJ BAZY:":INPUT">";NB$
1065 NB$=LEFT$(NB$,15)
1070 PRINT:INPUT"PODAJ LICZBE KLUCZY (1-10):";KL:KL=INT(ABS(KL)
1080 IF KL<1 OR KL>10 THEN1070
1090 PRINT:PRINTL$:DIMB$(300,KL):IL=0
1100 FOR T=1TOKL:PRINT"KLUCZ NR";T;":";:INPUT KL$(T):PRINT
1105 KL$(T)=LEFT$(KL$(T),16):NEXT
1110 PRINTCHR$(147);L$;X$;"NAZWA BAZY:";Y$;" ";NB$:PRINTL$;
1120 FOR T=1TOKL:PRINTX$;"KLUCZ NR";T;CHR$(157);":";Y$;KL$(T):NEXT:POKE19,0
1130 PRINT:PRINT"CZY WSZYSTKO W PORZADKU (T/N)?"
1140 GOSUB900
1150 IF A$="N"THEN1050
1160 GOTO 20
2000 IF IL>0 THEN2100
2010 PRINTCHR$(147);L$;SPC(4);"WPROWADZANIE NOWEGO ZAPISU NR";IL+1:PRINTL$
2020 IF IL=300 THEN PRINT"BAZA PELNA!!":FORT=1TO1500:NEXT:GOTO2000
2030 FOR T=1TO KL:PRINTX$;T;CHR$(157);".";KL$(T);Y$;TAB(19);
2040 POKE19,16:INPUT":";B$:POKE19,0:B$(IL+1,T)=LEFT$(B$,19):B$="":PRINT:NEXT
2050 PRINTCHR$(147);L$;SPC(4);"PRZYJETO DANE DLA ZAPISU NR";IL+1;":":PRINTL$
2060 FORT=1TO KL:PRINTX$;T;CHR$(157);".";KL$(T);Y$;TAB(19);":";B$(IL+1,T)
2070 NEXT:PRINT:PRINT"CZY WSZYSTKO SIE ZGADZA (T/N)?":GOSUB 900
2080 IF A$="N" THEN2010
2090 IL=IL+1:PRINT:PRINT"NASTEPNY ZAPIS (T/N)?":GOSUB900:IFA$="T"THEN2010
2100 K=IL
2101 GOSUB 2110:GOTO 2150
2110 POKE53265,11
2111 A$="BAZA DANYCH:"+NB$:C=19-INT(LEN(A$)/2):PRINTCHR$(147);L$;SPC(C);A$
2120 PRINTL$;"ZAPIS NR";K;":":PRINT
2130 FORT=1TO KL:PRINTX$;T;CHR$(157);".";KL$(T);Y$;TAB(19);":";B$(K,T):NEXT
2140 PRINT:PRINTL$;:POKE53265,27:RETURN
2150 PRINTX$;"[1]";Y$;"-WYSZUKIWANIE ZAPISU W/G KLUCZA"
2160 PRINTX$;"[2]";Y$;"-KASOWANIE ZAPISU"
2170 PRINTX$;"[3]";Y$;"-MODYFIKACJA ZAPISU"
2180 PRINTX$;"[4]";Y$;"-DOPISANIE NOWEGO ZAPISU"
2190 PRINTX$;"+/-";Y$;"-PRZEGLADANIE ZAPISOW"
```

64

```
2195 PRINTX$;"[M]";Y$;"-POWROT DO GLOWNEGO MENU"
2200 GETA$:IF A$="M" THEN 20
2210 IFA$="+" AND K<IL THEN K=K+1:GOTO 2101
2220 IFA$="-" AND K>1 THEN K=K-1:GOTO 2101
2230 IFA$="4" THEN 2010
2240 IFA$="2" THENPRINT:PRINT"NA PEWNO KASOWAC (T/N)?":GOTO 800
2250 IFA$="3"THEN 850
2260 IFA$<>"1"THEN2200
2270 PRINTCHR$(147);TAB(4);"WYSZUKIWANIE ZAPISOW W/G KLUCZY":PRINTL$
2280 FORT=1TOKL:PRINTT;CHR$(157);".";KL$(T):NEXT:PRINT:PRINT"PODAJ NUMER KLUCZA
2290 PRINT"(1 -";KL;CHR$(157);")";:INPUT W%:IFW%<1 OR W%>KL THEN 2270
2300 PRINT:POKE19,16:INPUT"PODAJ SZUKANY TEKST:";T$:POKE19,0
2310 K=1:M=1:N=1
2320 IFMID$(T$,N,1)="?"THEN2340
2330 IFMID$(T$,N,1)<>MID$(B$(M,W%),N,1)THEN 2360
2340 N=N+1:IFN<=LEN(T$)THEN2320
2350 K=M:GOSUB 2110:PRINT"NASTEPNY ZAPIS (T/N)?":GOSUB900:IFA$="N"THEN2370
2360 M=M+1:N=1:IFM<=ILTHEN2320
2370 PRINT:PRINT"KONIEC DANYCH DO PRZESZUKIWANIA.":FORT=1TO1900:NEXT:GOTO2101
3000 PRINTCHR$(147);L$;SPC(4);"ZAPISANIE BAZY NA NOSNIK ";D$:PRINTL$;
3010 PRINT"[1] - ZMIANA URZADZENIA (TASMA/DYSK)"
3020 PRINT"[2] - ZAPISANIE BAZY":IFPEEK(50000)=8THENPRINT"[3] - KATALOG DYSKU"
3025 PRINT"[M] - POWROT DO MENU"
3030 GETA$:IF A$="M" THEN20
3040 IF A$="2" THEN3090
3044 IFA$="3"AND PEEK(50000)=8THEN GOSUB6000:GOTO3000
3050 IF A$<>"1"THEN3030
3060 IF PEEK(50000)=8THEN X=1:GOTO3080
3070 X=8
3080 POKE50000,X:GOSUB5:GOTO3000
3090 PRINT:POKE19,16:INPUT"PODAJ NAZWE:";N$:POKE19,0:PRINT
3095 PRINT:PRINT"PRZYGOTUJ NOSNIK ";D$;".":PRINT"WCISNIJ [RETURN] GDY JUZ."
3100 GETA$:IF A$<>CHR$(13)THEN3100
3105 PRINT"OK..."
3110 IFPEEK(50000)=8THENOPEN1,8,15,"S:"+N$:CLOSE1
3120 OPEN1,PEEK(50000),1,N$+",S":PRINT#1,NB$:PRINT#1,IL:PRINT#1,KL
3130 FORT=1TOKL:PRINT#1,KL$(T):NEXT
3140 FORT=1TOIL:FORR=1TOKL:PRINT#1,B$(T,R):NEXT:NEXT:CLOSE1:GOTO20
4000 PRINTCHR$(147);L$;SPC(5);"ODCZYT BAZY Z NOSNIKA ";D$:PRINTL$;
4010 PRINT"[1] - ZMIANA URZADZENIA (TASMA/DYSK)"
4020 PRINT"[2] - ODCZYTANIE BAZY":IFPEEK(50000)=8THENPRINT"[3] - KATALOG DYSKU"
4025 PRINT"[M] - POWROT DO MENU"
4030 GETA$:IF A$="M" THEN20
4040 IF A$="2" THEN4090
4044 IFA$="3"AND PEEK(50000)=8THEN GOSUB6000:GOTO4000
4050 IF A$<>"1"THEN4030
4060 IF PEEK(50000)=8THEN X=1:GOTO4080
4070 X=8
4080 POKE50000,X:GOSUB5:GOTO4000
4085 CLR:GOSUB5
4090 PRINT:POKE19,16:INPUT"PODAJ NAZWE:";N$:POKE19,0:PRINT
4095 PRINT:PRINT"PRZYGOTUJ NOSNIK ";D$;".":PRINT"WCISNIJ [RETURN] GDY JUZ."
4100 GETA$:IF A$<>CHR$(13)THEN4100
4105 PRINT"OK..."
4120 OPEN1,PEEK(50000),0,N$+",S":INPUT#1,NB$:INPUT#1,IL:INPUT#1,KL
4130 FORT=1TOKL:INPUT#1,KL$(T):NEXT
4135 DIMB$(300,KL)
4140 FORT=1TOIL:FORR=1TOKL:INPUT#1,B$(T,R):NEXT:NEXT:CLOSE1:GOTO20
5000 PRINTCHR$(147);L$;SPC(6);"SORTOWANIE DANYCH W/G KLUCZA":PRINTL$
5010 FORT=1TOKL:PRINTT;CHR$(157);".";KL$(T):NEXT:PRINT
5020 PRINT"PODAJ NUMER KLUCZA, W/G KTOREGO ODBEDZIESIE SORTOWANIE ZAPISOW:"
5030 PRINT:PRINT"KLUCZ NR (1 -";KL;CHR$(157);"):";:POKE19,16:INPUTW%:POKE19,0
5040 PRINT:PRINT:PRINT"PROSZE POCZEKAC - TRWA SORTOWANIE..."
5050 IF IL=1THEN20
5060 FOR T=1TOIL-1:POKE211,0:PRINTIL-T;CHR$(157);"   ";:FORR=T+1TOIL
5065 IFB$(T,W%)<=B$(R,W%)THEN5080
5070 FORC=1TOKL:A$=B$(T,C):B$(T,C)=B$(R,C):B$(R,C)=A$:NEXT
5080 NEXT:NEXT:K=1:GOTO2101
6000 PRINT:OPEN1,8,0,"$":GET#1,A$,A$
6010 GET#1,T$,S$:GETA$:IF ST OR A$=" "THEN6050
6020 GET#1,A$,B$:PRINT TAB(5);ASC(A$+CHR$(0))+256*ASC(B$+CHR$(0));
6030 GET#1,A$:PRINTA$;:IF A$="" THEN PRINT:GOTO6010
6040 GOTO6030
6050 CLOSE1:PRINT:PRINTX$;"WCISNIJ DOWOLNY KLAWISZ...":WAIT198,1:RETURN
READY.
```

10...
20
30

Listing nr 2

```
/* Tabela kodow ASCII */
LF='0A'X
ADDRESS 'rexx_ced'
Wypisz='Tablica kodow ASCII'||LF||LF
DO i=32 TO 127
   Wypisz=Wypisz||CENTER(i,4)||D2C(i)
   IF i//8=7 THEN
      Wypisz=Wypisz||LF
   ELSE
      Wypisz=Wypisz||'    '
END
Okay1 Wypisz
```

Listing nr 4

```
/* Podreczny kalkulator */
OPTIONS RESULTS
SIGNAL ON SYNTAX
ADDRESS 'rexx_ced'
GetString
Wyr=RESULT
IF Wyr~='RESULT' & Wyr~='' THEN
DO
   Oblicz='a='||Wyr
   INTERPRET Oblicz
   Okay1 a
END
EXIT 0
SYNTAX:
   Okay1 'Wyrazenie jest niepoprawne'
EXIT 0
```

Listing nr 3

```
/* Tabela kodow ASCII */
LF='0A'X
NazwaTablicy='ASCII.clip'
ADDRESS 'rexx_ced'
Wypisz=GETCLIP(NazwaTablicy)
IF Wypisz='' THEN DO
   Wypisz='Tablica kodow ASCII'||LF||LF
   DO i=32 TO 127
      Wypisz=Wypisz||CENTER(i,4)||D2C(i)
      IF i//8=7 THEN
         Wypisz=Wypisz||LF
      ELSE
         Wypisz=Wypisz||'    '
   END
   CALL SETCLIP(NazwaTablicy,Wypisz)
END
Okay1 Wypisz
```

Listing nr 5

```
/* Slownik w ARexx'ie */
SIGNAL ON ERROR
LF='0A'X
Slownik='Dictionary.txt'
OPTIONS RESULTS
ADDRESS 'rexx_ced'
Status 21        /* Wez nazwe aktualnego pliku z Cygnus'a */
AktualnyPlik=UPPER(RESULT)
'Jump to file' UPPER(Slownik)      /* Przenies kursor do odpowiedniego pliku */
IF ~RESULT THEN
   CALL Wyjdz 'Nie moge znalezc slownika!'
/* Wywolaj procedure Wyjdz z parametrem 'Nie mog znale slownika */
'GetString'
DoTlumaczenia=RESULT
IF (DoTlumaczenia=='') | (DoTlumaczenia='RESULT') THEN
   CALL Wyjdz 'Tlumaczenie anulowane'
'Beg of file'      /* Skocz na poczatek slownika */
'Search for' DoTlumaczenia 1 0 1 1       /* Szukaj wyrazu */
IF RESULT~=0 THEN
DO    Status 55        /* Wez linie tekstu z Cygnus'a */
   Przetlumaczone=RESULT
   CALL Wyjdz COMPRESS(Przetlumaczone,LF)
/* COMPRESS(tekst,zn) usuwa z <tekst> wszystkie wystapienia znaku <zn> */
END
GetString
Opis=RESULT
IF (Opis=='') | (Opis='RESULT') THEN
   CALL Wyjdz 'Opis anulowany'
'End of file'      /* Te trży linie dopisuja nowe okreslenie */
Text DoTlumaczenia||' - '||Opis||LF
CALL Wyjdz 'Wprowadzono nowe slowo'

Wyjdz:       /* A to nasza procedura Wyjdz */
```

```
   PARSE ARG Text
   'Jump to file' AktualnyPlik
   Okayl Text
EXIT 0

Error:
   Okayl ERRORTEXT(RC)||'0A'x||'in line '||SIGL
EXIT 0
```



Listing nr 6

```
"POLSKIE ZNAKI 64"
$0801-$0F54
----------------------------------------
:0801 0B 08 90 06 9E 32 30 34 (1E)
:0809 39 00 A0 00 78 E6 01 B9 (B5)
:0811 E1 0E 99 40 03 C8 D0 F7 (08)
:0819 4C 40 03 36 62 0E 0B 41 (61)
:0821 A0 04 C2 9C FF A0 A0 18 (62)
:0829 20 99 FF A4 B6 6F 6C 00 (2C)
:0831 A0 48 8A 48 98 48 A9 7F (66)
:0839 8D 5C 57 AC 0D DD 30 35 (A2)
:0841 20 BC F6 20 E1 FF D0 2D (BA)
:0849 20 A3 94 AF 15 FD 20 1B (3E)
:0851 E5 A9 94 8D 00 DD A9 17 (05)
:0859 E8 15 D0 A9 01 8D 86 AD (EC)
:0861 C4 8D 88 02 A9 68 8D 18 (3F)
:0869 03 A9 C0 8D 19 03 A2 FB (0F)
:0871 9A 4C AF C0 4C 72 FE 20 (D2)
:0879 CC FF 4C 74 A4 48 F1 05 (A2)
:0881 FE 83 03 FC 07 07 F8 0F (13)
:0889 0E F0 1F 1C E0 3F 38 C0 (AE)
:0891 7F 70 80 FF E0 00 FF C1 (D5)
:0899 01 FE 83 03 FC 07 07 F8 (3A)
:08A1 0F 0E 50 15 9C 99 6C 6C (0E)
:08A9 84 56 7B 88 08 AB 70 30 (2C)
:08B1 3E 09 33 4E 3D 19 E5 D7 (9C)
:08B9 FF 06 4C 66 3F 02 13 16 (C4)
:08C1 FE 03 1C 36 1A 78 C5 59 (76)
:08C9 96 91 24 2D AB E0 07 62 (81)
:08D1 22 2A 11 03 1C F8 A7 6C (DB)
:08D9 38 E0 60 66 6C 78 64 43 (51)
:08E1 38 32 D8 42 EC A2 25 76 (30)
:08E9 FC D4 70 E4 40 F1 78 BE (93)
:08F1 DC 06 7C FC A1 96 76 DC (08)
:08F9 CC 7C 94 63 EC 76 60 60 (0D)
:0901 F0 CA FA 00 68 41 9B 1E (37)
:0909 30 36 1C 79 60 C0 22 61 (3C)
:0911 3C 83 E6 A5 17 7E 6C C7 (35)
:0919 83 36 3C 6C B2 8C FC E1 (23)
:0921 7C F5 A5 4C 18 32 92 71 (D9)
:0929 01 30 D0 A3 0C 1A 30 7C (97)
:0931 30 62 FC D9 84 01 0C 74 (14)
:0939 54 70 2C 08 10 30 7F 7F (FB)
:0941 30 10 58 93 E1 AC 25 AB (B6)
:0949 58 13 30 82 FF 66 FF 4C (20)
:0951 D0 18 3E E0 FD 7C 8C 63 (59)
:0959 F7 53 50 80 AF 3C 3C 67 (9E)
:0961 66 3B 6C 35 CB 90 5A 80 (2B)
:0969 09 0C 00 40 AE 19 8E 12 (01)
:0971 A9 FF 7B 5B 4B 7E 5D 22 (04)
:0979 4F 30 36 CE 02 64 42 22 (4B)
:0981 63 7A 38 C0 F9 6E 76 07 (6C)
:0989 4C 18 38 78 A3 54 C3 51 (9A)
:0991 1C FF 4E 32 02 76 06 0E (CE)
:0999 1E 36 7F 06 06 65 47 21 (36)
:09A1 60 91 90 C8 08 CC 08 98 (E4)
:09A9 5E 8F 43 20 31 49 1E 3E (E4)
:09B1 03 26 07 A6 F1 A2 38 0E (2F)
:09B9 C4 47 97 18 FA 8B 93 CE (D2)
:09C1 6D 8E 70 74 D4 A5 18 70 (9D)
:09C9 8E 0F 06 1C A8 37 BC E9 (FE)
:09D1 9F 49 49 E2 F8 EA 84 7C (3E)
:09D9 7F 08 0B AC 32 62 5A FC (DE)
:09E1 7C 60 66 E7 FE 7D 29 61 (FF)
:09E9 60 F6 5E 29 31 4D CB 09 (94)
:09F1 9E 24 22 1E 0D 0C 30 D2 (27)
:09F9 2A 91 5E 71 C8 1C C8 FE (24)
:0A01 F2 23 EE FE D6 CE 82 C8 (D5)
:0A09 76 76 7E 6E 6E 5B 59 24 (7E)
:0A11 0D F3 60 60 10 26 2D 78 (DD)
:0A19 66 6C 36 00 A2 7B 7C 78 (33)
:0A21 6C C9 B8 92 87 E2 1E 7E (4A)
:0A29 5A 70 34 39 32 9F E6 F3 (83)
:0A31 56 02 00 3A C6 D6 FE EE (C1)
:0A39 C6 58 0F 30 15 00 CE 82 (C1)
:0A41 13 A5 24 23 90 62 12 60 (3A)
:0A49 30 30 31 47 75 27 25 25 (F0)
:0A51 A9 43 18 8B 06 0C C8 00 (DC)
:0A59 70 DB 0E 38 45 C3 B4 52 (FA)
:0A61 8D 03 64 28 17 80 68 CD (7D)
:0A69 00 19 8C C5 D4 7E 66 0C (9F)
:0A71 7E 30 2A 09 40 03 36 8B (1F)
:0A79 00 33 33 F0 60 4E 85 E2 (A9)
:0A81 AF 10 FC 25 02 66 00 20 (50)
:0A89 1F 1F 01 C6 61 44 A2 18 (B6)
:0A91 00 12 E6 C4 E4 3E 60 8C (69)
:0A99 A1 7C 03 98 B9 3C 06 3E (C4)
:0AA1 66 3F 06 C2 2E 05 7E 60 (1F)
:0AA9 3E 0C 4C CD 0A 50 18 F8 (9B)
:0AB1 F8 09 18 0E 64 C0 03 E0 (CE)
:0AB9 D9 1C 18 38 63 9E 3A 00 (35)
:0AC1 00 D4 00 01 0C 18 2A 5C (49)
:0AC9 66 9D 01 03 06 6C 78 70 (F0)
:0AD1 60 44 16 20 A0 18 10 3C (85)
:0AD9 66 60 66 3C 00 0C 08 7E (9B)
:0AE1 4C 18 32 7E 92 EF 51 00 (A0)
:0AE9 00 06 F0 18 0F E0 99 93 (81)
:0AF1 93 A0 D5 1E 22 C2 8F CF (5C)
:0AF9 C1 6A 82 CC 53 4F 46 F9 (79)
:0B01 F5 3F F9 93 99 C0 C0 84 (93)
:0B09 85 FF 80 E3 C9 46 87 71 (6D)
:0B11 96 65 24 49 CB 2A F8 81 (CF)
:0B19 98 88 4A C4 00 E3 FE 29 (46)
:0B21 93 C7 1F 9F 99 93 87 D9 (0E)
:0B29 90 C7 0C B6 90 13 68 89 (B0)
:0B31 1D 03 35 1C 39 50 3C 9E (FF)
:0B39 23 AF 01 83 7F A8 89 23 (16)
:0B41 33 83 25 E5 98 13 89 9F (A9)
:0B49 9F 0F B2 3E 00 5A D0 A6 (1B)
:0B51 CF C9 E3 47 1E 18 B0 48 (B8)
:0B59 C3 D8 60 19 E9 C5 81 93 (A5)
:0B61 F1 C9 C3 93 A0 2C 23 7F (99)
:0B69 78 83 7D A9 B3 E7 CD 64 (B1)
:0B71 5C 00 CF F4 F3 E5 CF 83 (F3)
:0B79 CF 9D 03 68 36 61 00 F3 (22)
:0B81 1D 15 1C 0B EF CF 80 80 (58)
:0B89 CF EF 02 D6 64 38 6B C9 (18)
:0B91 2A D6 04 8C 20 00 99 00 (7D)
:0B99 13 34 E7 C1 78 3F 83 E3 (57)
:0BA1 9C FD 14 14 E0 2B C3 C3 (9D)
:0BA9 98 99 C4 5B CD 32 A4 16 (8F)
:0BB1 60 02 F3 FF 90 6B 86 A3 (09)
```

```
:0BB9 44 EA 00 DE D6 52 81 97 (AD)
:0BC1 C8 93 CF 8D B3 00 99 90 (89)
:0BC9 C8 98 1E 4E 3F FE 91 89 (CC)
:0BD1 01 D3 E7 C7 87 28 D5 70 (3A)
:0BD9 E3 D4 BF 93 8C 80 5D F9 (07)
:0BE1 F1 E1 C9 80 F9 F9 D9 51 (24)
:0BE9 08 58 24 24 32 02 33 02 (23)
:0BF1 A6 D7 E3 10 48 4C 92 C1 (6F)
:0BF9 C7 80 C9 81 69 BC 28 0E (27)
:0C01 F1 F1 D1 A5 E7 FE E2 A4 (AC)
:0C09 73 9B 23 8F 1D 75 A9 E7 (89)
:0C11 8F E3 F9 E3 03 EA 0D 6F (47)
:0C19 FA 67 52 92 B8 07 3A 83 (9B)
:0C21 E1 1F C2 02 AB 8C 98 16 (11)
:0C29 03 1F 99 D8 79 01 5F 4A (ED)
:0C31 98 9F BD 57 4A 4C D3 72 (35)
:0C39 82 27 89 48 E1 03 F3 8C (4C)
:0C41 B4 4A A4 57 1C 32 07 B2 (56)
:0C49 01 FC 88 11 01 29 B3 20 (0A)
:0C51 F2 89 89 81 91 91 56 16 (45)
:0C59 49 C3 3C 9F 9F 84 49 0B (EE)
:0C61 9E 99 93 C9 FF E8 5E 83 (2F)
:0C69 87 93 32 AE E4 A1 B8 81 (BA)
:0C71 A5 07 1C 4D 8E CC E7 19 (5F)
:0C79 BC 95 80 FF 0E 39 29 01 (AA)
:0C81 11 39 D6 03 4C 85 FF B3 (C9)
:0C89 E0 44 29 C9 08 A4 98 44 (E4)
:0C91 9F CF CF CC 8A D8 DA DA (FF)
:0C99 51 EA E7 D0 F9 F3 22 32 (CC)
:0CA1 FF 8F 24 F1 4E D1 30 AD (1E)
:0CA9 54 E3 00 19 CA 05 20 5A (F3)
:0CB1 33 40 06 63 31 81 99 F3 (D0)
:0CB9 81 CF B5 4A 02 90 FC CD (E1)
:0CC1 33 33 CC CC 0F 9F 22 AC (4D)
:0CC9 50 FC B5 EF 03 44 99 FF (C6)
:0CD1 00 24 E0 E0 C0 38 8C 08 (69)
:0CD9 F0 E7 FF 40 E6 C4 E4 C1 (FA)
:0CE1 9F 8C A1 83 03 98 B9 C3 (59)
:0CE9 F9 C1 99 C0 F9 C2 2E 05 (0E)
:0CF1 81 9F C1 F3 4C CD 0A 50 (DB)
:0CF9 E7 07 07 09 E7 0E 64 3F (BE)
:0D01 03 1F D9 E3 E7 C7 63 9E (38)
:0D09 3A 00 FF D4 00 E0 A0 F3 (D5)
:0D11 E7 C2 99 D5 49 FE FC F9 (B5)
:0D19 93 87 8F 9F 64 01 02 E7 (30)
:0D21 EF C3 99 9F 99 C3 FF F3 (0A)
:0D29 F7 81 B3 E7 CD 81 2A F9 (D9)
:0D31 1E 05 FF 60 0F 80 F0 AA (0E)
:0D39 41 C8 C8 C8 A9 00 85 10 (FF)
:0D41 A6 FB 30 0D 06 FA 2A 26 (FE)
:0D49 10 C6 FB 88 D0 F2 AA 18 (25)
:0D51 60 48 A1 FF 85 FA A2 07 (48)
:0D59 86 FB E4 FF A6 FE D0 02 (7C)
:0D61 C6 FF C6 FE 68 90 DD E0 (EF)
:0D69 E7 D0 D9 A9 37 85 01 58 (24)
:0D71 4C 00 C0 A9 10 85 04 AA (88)
:0D79 BC D8 07 20 32 07 A6 04 (55)
:0D81 7D C7 07 48 A5 10 69 00 (46)
:0D89 E0 0F 69 00 A8 68 A6 02 (21)
:0D91 D0 08 C0 00 D0 04 C9 01 (6D)
:0D99 F0 D9 18 65 FC AA 98 65 (5C)
:0DA1 FD 85 49 A4 11 F0 20 8A (45)
:0DA9 38 E5 11 B0 03 C6 49 38 (1D)
:0DB1 85 48 A5 FC E5 11 B0 02 (71)
:0DB9 C6 FD 85 FC B1 48 88 91 (6A)
:0DC1 FC D0 F9 C4 12 F0 89 B1 (A6)
:0DC9 48 C6 FD C6 49 C6 12 10 (C8)
:0DD1 ED 00 04 0C 1E 3E 7F CE (FA)
:0DD9 FF 00 08 13 26 46 86 CF (CD)
:0DE1 00 80 02 03 04 05 06 06 (8C)
:0DE9 06 07 03 03 04 05 06 06 (AB)
:0DF1 06 07 07 97 11 9A 18 E5 (44)
:0DF9 78 A0 00 84 FB A9 D0 85 (83)
:0E01 FC A9 34 19 7C 53 A2 08 (69)
:0E09 C6 01 B1 FB E6 01 91 FB (31)
:0E11 C8 D0 F5 E6 FC CA D0 F0 (D6)
:0E19 A9 37 85 01 F3 1B 31 00 (89)
:0E21 8D 20 D0 8D 21 1C 88 02 (B5)
:0E29 00 E0 39 0D B9 27 0D 99 (80)
:0E31 1C 07 C8 D0 F7 20 31 07 (23)
:0E39 F0 46 20 31 07 D0 30 20 (3A)
:0E41 30 07 69 02 C9 04 90 27 (FD)
:0E49 D0 07 20 31 07 69 04 D0 (8E)
:0E51 1E 20 2F 07 69 06 C9 0D (7E)
:0E59 D0 11 C8 20 2F 07 69 0D (8D)
:0E61 C9 0E D0 07 A0 01 20 32 (76)
:0E69 07 69 1D EE 00 04 EA 85 (05)
:0E71 11 A6 FE A5 FF 20 97 07 (7E)
:0E79 A5 49 85 FF A5 48 85 FE (C5)
:0E81 20 31 07 85 02 F0 14 0A (C0)
:0E89 2C A9 03 85 11 20 2F 07 (C8)
:0E91 A6 02 D0 02 69 08 20 6B (36)
:0E99 07 F0 9A 20 31 07 F0 E9 (D3)
:0EA1 20 30 07 69 04 C9 06 90 (5C)
:0EA9 E2 D0 07 20 30 07 69 06 (F7)
:0EB1 D0 D9 A0 05 20 32 07 69 (7A)
:0EB9 0A C9 0A F0 0D C9 0C D0 (0C)
:0EC1 CA A0 04 20 32 07 69 FF (60)
:0EC9 D0 C1 A0 09 20 32 07 A6 (5A)
:0ED1 10 86 12 90 B6 A9 00 85 (1D)
:0ED9 2D A9 E0 85 2E 4C 00 C0 (C8)
:0EE1 C6 01 58 A7 A2 CD 12 D0 (51)
:0EE9 D0 34 A2 06 A9 FF 95 F8 (47)
:0EF1 BD 90 03 95 F7 CA CA 10 (CE)
:0EF9 F3 A2 12 BD A0 03 4D 5E (A5)
:0F01 03 9D E8 07 CA 10 F4 A9 (67)
:0F09 13 A2 E8 A0 07 20 97 03 (C3)
:0F11 A9 0F A8 A2 08 20 94 03 (73)
:0F19 20 9A 03 20 9D 03 29 00 (47)
:0F21 A8 78 E6 01 B9 27 0E 99 (2F)
:0F29 FA 00 C8 D0 F7 4C 00 01 (6D)
:0F31 BA BD C0 E7 6C F7 00 6C (96)
:0F39 F9 00 6C FB 00 6C FD 00 (E4)
:0F41 4D 7D 77 0E 02 01 1A 14 (9A)
:0F49 03 62 19 05 08 6D 0B 02 (91)
:0F51 1F 0E 08 09 03 08 6D 60 (11)
```

Listing nr 7

```
0 REM ZARLOCZNY ROBACZEK - GRA
1 REM (C) PAWEL SOLTYSINSKI
2 REM 1992 COMMODORE KEBAB
3 :
5 DIM AD%(39,24):PRINTCHR$(147);"PROSZE CHWILE POCZEKAC...":DIM OG%(1000)
10 DEF FN A(X)=1024+40*Y+X
20 FOR X=0TO39:FORY=0TO24:AD%(X,Y)=FNA(X):NEXT:NEXT
30 DIM J%(15):J(1)=1:J(2)=2:J(4)=3:J(8)=4
40 POKE53280,0:POKE53281,0:PRINTCHR$(5);CHR$(147);CHR$(14);CHR$(8);
50 FOR X=0TO39:POKEAD%(X,1),95:POKEAD%(X,24),95:NEXT
60 FORY=1TO23:POKEAD%(0,Y),95:POKEAD%(39,Y),95:NEXT
```

```
70 PRINTTAB(9);"WCISNIJ GUZIK NA JOY#2"
80 WAIT 56320,16,16:FORT=0TO39:POKEAD%(T,0),32:NEXT
90 A=0:B=1:OG%(0)=AD%(19,10):OG%(1)=AD%(19,11):S1=0:S2=1:X=19:Y=11:P=1:SC=-1
95 FORT=0TO39:POKEAD%(T,0),32:NEXT:PRINTCHR$(19);"WYNIK:";0:O1=X:O2=Y
100 IF X=O1 AND Y=O2 THENSC=SC+1:PRINTCHR$(19);TAB(6);SC:GOSUB200:P=0
110 I=PEEK(OG%(S2)):IFI=37 OR I=95 THEN300
115 POKE OG%(S2),35
120 J=15-(PEEK(56320)AND15):IFJ=0THEN130
125 ON J(J) GOSUB 400,500,600,700
130 IFP=0THEN 140
135 S1=S1+1:IFS1=1000 THENS1=0
140 P=1:POKE(OG%(S2)),37:S2=S2+1:IFS2=1000THENS2=0
145 X=X+A:Y=Y+B:OG%(S2)=AD%(X,Y)
150 POKE OG%(S1),32
160 GOTO 100
200 O1=INT(RND(TI)*37+1):O2=INT(RND(TI)*21+2)
210 IF PEEK(AD%(O1,O2))<>32THEN200
220 POKE53280,2:POKE AD%(O1,O2),79:POKE53280,0
240 B1=INT(RND(TI)*37+1):B2=INT(RND(TI)*21+2)
250 IF PEEK(AD%(B1,B2))<>32THEN240
260 POKE AD%(B1,B2),95:RETURN
300 POKE OG%(S2),35:FORT=255TO0STEP-1:POKE53280,T:NEXT:PRINTCHR$(19);TAB(15);
310 PRINT"GAME OVER!":WAIT 56320,16,16:GOTO40
400 A=0:B=-1:RETURN
500 A=0:B=1:RETURN
600 A=-1:B=0:RETURN
700 A=1:B=0:RETURN
READY.
```

Listing nr 8

```
A5000 LDA #$00
A5002 STA $D020
A5005 STA $D021
A5008 LDA #$93
A500A JSR $FFD2
A500D LDX #$00
A500F LDA $5080,X
A5012 BEQ $501A
A5014 JSR $FFD2
A5017 INX
A5018 BNE $500F
A501A LDX #$00
A501C JSR $FFCF
A501F CMP #$0D
A5021 BEQ $5029
A5023 STA $5100,X
A5026 INX
A5027 BNE $501C
A5029 TAY
A502A LDA #$00
A502C STA $5100,X
A502F TYA
A5030 JSR $FFD2
A5033 JSR $FFD2
A5036 JSR $FFD2
A5039 LDX #$00
A503B LDA $5098,X
A503E BEQ 5046
A5040 JSR $FFD2
A5043 INX
A5044 BNE $503B
A5046 LDX #$00
A5048 LDA $5100,X
A504B BEQ $5053
A504D JSR $FFD2
A5050 INX
A5051 BNE $5048
A5053 LDA #$3F
A5055 JSR $FFD2
A5058 LDA #$20
A505A JSR $FFD2
A505D JSR $FFD2
A5060 LDX #$00
A5062 LDA $50A0,X
A5065 BEQ $506D
A5067 JSR $FFD2
A506A INX
A506B BNE $5062
A506D JSR $FFE4
A5070 BEQ $506D
A5072 RTS

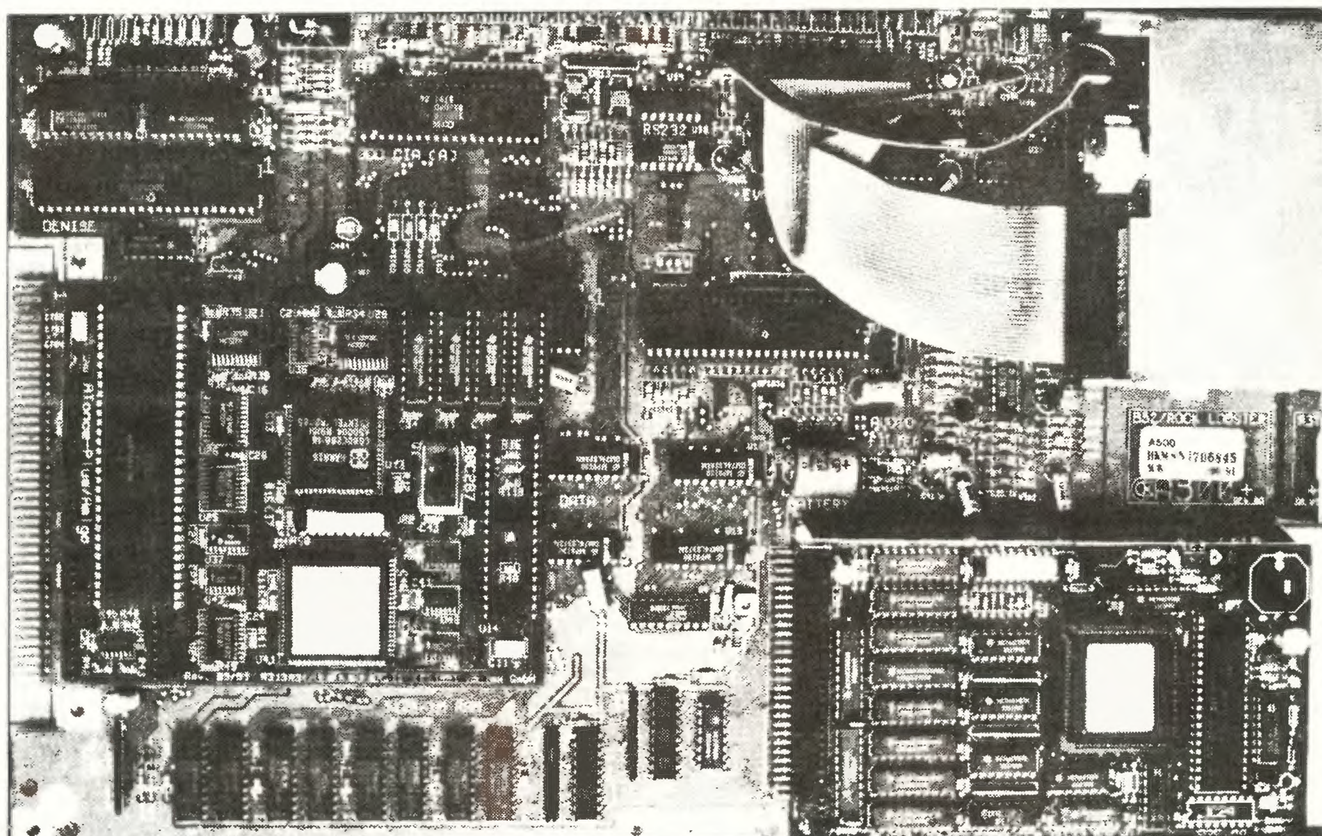
M5080 4A 41 4B 20 4D 41 53 5A
M5088 20 4E 41 20 49 4D 49 45
M5090 20 3F 20 20 00 BD BD BD
M5098 4A 41 4B 20 3F 20 20 00
M50A0 4E 49 45 42 52 5A 59 44
M50A8 4B 4F 2E 2E 2E 21 00 BD
```

"Listów" ciąg dalszy

Drodzy Czytelnicy,
ze smutkiem przyjęliśmy fakt, że pan Krzysztof Moroń nie był uprzejmy dokończyć rozpoczętego dzieła. Niemniej jednak mamy w zamiarach powrócenie do tematu AMOS natychmiast, gdy w gronie redakcyjnym znajdzie się odpowiedzialna osoba będąca w stanie poprowadzić cykl artykułów dotyczących tego doskonałego i zdobywającego coraz więcej zwolenników języka programowania. Wtedy również odpowiemy na pozostałe pytania dotyczące AMOS'a zawarte w cytowanych powyżej, oraz wielu innych listach.

Cześć Kebabie!
Kto wam tak "spartaczył" majowy numer? Niektóre wyrazy nie dają się w ogóle odczytać!...

Odpowiedź: profesjonaliści!
Ten numer składamy ponownie po amatorsku sami... Mamy nadzieję, że da się przeczytać...



To zdjęcie miało być w numerze 5'92 - dwa "pecety" w Amidze



Kupon ogłoszeniowy

----
imię i nazwisko

----
adres

treść:

